The Signal in the Noise with Machine Learning   
Algorithms and Spine Classification

Michael Streyle

November 14, 2018

Machine learning is commonly thought of as a tool primarily used in data science, however, machine learning is also very much a part of applied statistics. Most, if not all, machine learning algorithms were first developed and used by statisticians. Inspired by the book “The Signal and the Noise: Why So Many Predictions Fail-But Some Don't”, by Nate Silver, the main goal of this project is to use a relatively small data set and compare how various supervised machine learning algorithms find the signal in the data while incrementally adding random noise. The data set that I have chosen for the analysis consists of twelve spinal measurements and a classification variable consisting of two factor levels, normal and abnormal. The algorithms I am going to use to classify the spinal measurements include logistic regression, support vector machines, random forests, and an algorithm called “Generalized, Unbiased, Interaction Detection and Estimation” (GUIDE), developed by Wei-Yin Loh at the University of Wisconsin, Madison. First, I compare the algorithms on the regular data before adding 10, 100, 500, and 1000 variables of random noise and comparing the performance of the various models. The detailed output of logistic regression can be found in Appendix II, support vector machines in Appendix III, random forests in Appendix IV, and GUIDE tree diagrams in Appendix V.

The spinal measurements I am going to use in this analysis are publicly available on Kaggle. The objective is to classify each observation as normal or abnormal using machine learning modeling. As seen in the pairs plot from Appendix I, none of the variables have strong correlations with each other, and the last 6 variables are essentially random noise. These random variables will be left out of some of the initial models so that I can compare the models with only meaningful variables with the models that include random variables in order to assess how the models handle excess noise in the data. The six non-random variables (pelvic\_incidence, pelvic\_tilt, lumbar\_lordosis\_angle, sacral\_slope, pelvic\_radius, and degree\_spondylolisthesis) all appear relatively normal and symmetric except degree\_spondylolisthesis, which appears be right skewed.

Logistic regression is one of the most popular and most powerful classification algorithms. Logistic regression is essentially linear regression on the logit scale so that all the predictions are between 0 and 1. The logit scale is the natural log of the odds that the response is one of the categories. Then, a cutoff, such as 0.5, is chosen and used to classify each observation as one of the levels of the response variable. In R, I use the glm function to create and train a logistic regression model with spine classification (Abnormal, Normal) as the response variable and the provided real spinal measurements as the predictor variables. By splitting the data into a training and test set, I am training the model on a randomly selected 80% subset of the data and keeping 20% of the data to test the performance of the model. After training the model, the summary output shows that the most important variables in the model are pelvic\_radius and degree\_spondylolisthesis with respective p-values of 0.0028 and , which are both well below any reasonable alpha value. The results of the train/test logistic regression with only the six real variables is an accuracy score of 0.8387 with 36 true positives, 6 false positives, 16 true negatives, and 4 false negatives, with positives representing abnormal classification.

Next, I added the two-way interaction between the two most significant variables: pelvic\_radius and degree\_spondylolisthesis. The resulting model is very similar to the initial first-order model. The summary of this interaction model indicates the interaction is not significant with a p-value of 0.131, which is above most reasonable alpha cutoff values, although it is close and could be included. In order to test whether the interaction is significant, I then computed the confusion matrix and the accuracy score for a model with the two-way interaction and compared the results with the first-order model without the interaction. The interaction model has an accuracy score of 0.8225, with 36 true positives, 7 false positives, 15 true negatives, and 4 false negatives. The results show that the interaction made the model worse; the model with the interaction got one more prediction wrong than the first order model. Specifically, the interaction model produced one more false negative than the first order model, which decreased the accuracy score from 0.8387 to 0.8225.

A common variable selection technique for logistic regression is a stepwise procedure which models the data with a few variables at a time in order to filter out any unneeded variables. The result of the step() function in R is a model with only variables that are statistically significant. In this analysis, I only use the forward stepwise procedure which starts with a model with only an intercept and incrementally adds variables. Using this method, I will still be able to use variable selection when I add many more random variables. The results of the forward stepwise procedure are an accuracy score of 0.8548, with 37 true positives, 6 false positives, 16 true negatives, and 3 false negatives, again with positives representing abnormal classification. The variables included in this forward stepwise model are degree\_spondylolisthesis (p-value = 2.98e-11), sacral\_slope (p-value = 1.15e-05, pelvic\_radius (p-value = 0.00012), pelvic\_tilt (p-value = 0.01426), and Direct\_tilt (p-value = 0.09473). All of these variables have very low p-values, except Direct\_tilt which is only marginally significant.

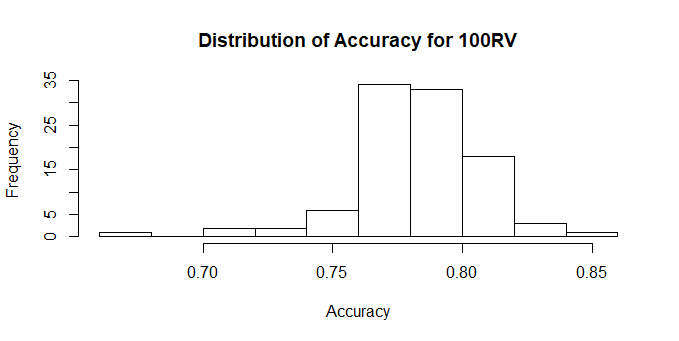
All of the initial models use a cutoff value of 0.5. This means that fitted probabilities above 0.5 are categorized as normal and fitted probabilities below 0.5 are categorized as abnormal. Instead of using 0.5 as the cutoff value, it is better to test various cutoff values in order to optimize the model's performance. The code used to find the optimal cutoffs uses the fitted probabilities of the training set as possible cutoffs, and computes the accuracy score for the training set using each fitted probability as a cutoff. The cutoff with the highest accuracy score on the test set is then used as the optimal cutoff. Using this method, the optimal cutoff for the first order logistic regression model is 0.7252. With this optimal cutoff and using the same forward stepwise method as above, the train/test split model’s accuracy score actually decreases from 0.8548 to 0.7419. This decrease in accuracy score highlights the main weakness of the train/test split methodology. The model is only tested on 20 percent of the data, which can lead to the model overfitting on the training set and performing poorly on the test set. In other words, the accuracy score optimizing method over-optimizes the model for the training set which then weakens the accuracy score of the test set predictions. An improved method to the train/test method is a procedure called cross-validation.

Cross-validation is a widely used method which enables the model to make predictions for the whole dataset. To do this, the data is split into n fold, or subgroups. Then the model is trained n times, each time leaving out one of the subgroups and using it as a test set. The result is predictions for every row in the data, yet still using separate data to train each model than what is used to test the model. Here, the confusion matrix and accuracy score is indicative of the model being tested on every data value, because the values of the confusion matrix add up to 310 (the number of observations in the data) rather than 62 (20 percent of the observations in the data).

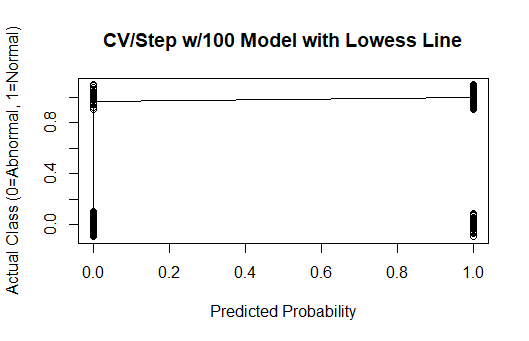
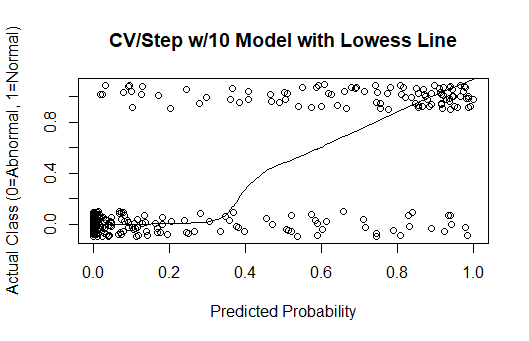
In this analysis, I implement cross-validation manually by making 5 subcategories of the data and training the model 5 times, each time keeping one fifth of the data out of training to be used as a test set. For the rest of the logistic regression models, I continue to use cross-validation, the forward stepwise variable selection method, and the tuning method to optimize the performance. All of these steps are performed while iterating through the code 5 times, once for each fold of the cross-validation. The logistic regression model trained still only on the 6 real variables with cross-validation, stepwise procedure, and optimization has an accuracy score of 0.8581 with 189 true positives, 23 false positives, 77 true negatives, and 21 false negatives with positives representing abnormal and negatives representing normal classifications. Not only does this model perform slightly better than the previous models, but it is much more conclusive and less prone to overfitting since it utilizes all the data to make test predictions.

One of the main goals of this project is to assess how the chosen models perform as the amount of noise in the data increases. The first step towards this goal is to add in the noise that was provided with the data. By adding the additional six random variables, and comparing to the previous logistic regression models, I can determine how logistic regression handles a small amount of increased random noise before continuing to even more random noise. The result of the logistic regression model using cross-validation, forward stepwise variable selection, and optimizing the results is an accuracy score of 0.8548 with 193 true positives, 28 false positives, 72 true negatives, and 17 false negatives. The significant variables in this model are degree\_spondylolisthesis, pelvic\_radius, sacral\_slope, and pelvic\_tilt, all with p-values below 0.02. Considering the number of variables doubled without any additional meaningful data, this model maintains a relatively high performance.

Next, I add an additional 10 random variables to the logistic regression model. These random variables were drawn from a standard normal distribution and are named X1, X2, … X10, respectively. After training a logistic regression model with cross-validation, forward stepwise variable selection, and performance optimizing, the resulting model still has degree\_spondylolisthesis, sacral\_slope, pelvic\_radius, and pelvic\_tilt as the most significant variables. Because cross-validation, stepwise variable selection, and performance optimizing all take place in the same loop, the only model summary I can easily access is the fifth iteration, however, this model can serve as an approximation of what the overall model looks like. In addition to the real variables, the model also has identified X1 as marginally significant. This indicates that logistic regression starts to break down with the addition of 16 random variables (6 provided and 10 created) to the 6 actual spine measurements. Despite the two marginally significant random variables, the model still performed reasonably well. The accuracy score is 0.8419 with 187 true positives, 26 false positives, 74 true negatives, and 23 false negatives, again with positives representing abnormal classification.

Next, I add 100 random variables to the 12 provided spinal measurement, again drawn from the standard normal distribution. The same process as above is used to train the model, and results in an accuracy score of 0.7774, with 175 true positives, 34 false positives, 66 true negatives, and 35 false negatives. The additional 90 random variables has decreased the accuracy score from 0.8419 to 0.7774, which is the result of both fewer correct abnormal classification and fewer correct normal classifications. Below is a histogram of the accuracy score after training the model with the 100 random variables 100 times. The histogram gives insight into the distribution of the accuracy of this model, and confirms that the model would continue to perform with an accuracy score between 0.75 and 0.82. 

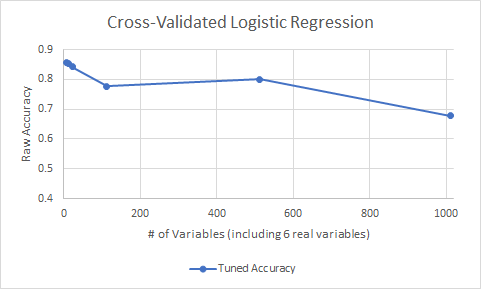
For each logistic regression model, I plotted the fitted probabilities against the actual classification with a lowess line fit to the data. The resulting plot reveals the relation between the predicted probability of the model and the actual class. Comparing two of these plots shows how logistic regression breaks down as the variables increase. Below is the plot for the model with 10 random variables and the plot for the model with 100 random variables. The plot for the 100 random variables shows quite clearly that the model has stopped producing meaningful fitted probabilities and is predicting 0’s and 1’s. This is concerning because it is impossible for the model to have predicted probabilities of complete certainty.



Another method for dealing with excess noise or too many extra variables is dimension reduction, and I implemented Principal Component Analysis to try to reduce the dimensions of the data. The concept behind Principal Component Analysis is to transform the data in such a way that the maximum amount of variation is captured in the smallest number of variables, by using linear combinations of the existing variables to create new, transformed variables called the principal components. The results of the PCA analysis are in Appendix II and indicate that this data will not nicely reduce its dimensionality as I had hoped. This is most likely due to the fact that 100 of the 113 variables are random noise. Thus, the variation is not able to be captured in a linear combination of the variables in such a way that reduces the dimensionality. Therefore, I conclude that the original, cross-validated logistic regression model with forward stepwise variable selection is the best I can do.

Similar to above, I tried to use Principal Component Analysis with the addition of 500 random variables. The results can also be found in Appendix II, and are even worse than with the 100 random variables. Here, it would require including approximately 150 principal components in order to capture only 80 percent of the cumulative variance in the data and there is no clear drop off in variability. The results of the forward stepwise procedure results in an accuracy score of 0.80 with 175 true positives, 27 false positives, 73 true negatives, and 35 false negatives. Considering there are only 6 real variables out of 512 total variables, logistic regression is performing quite well. The stepwise variable selection is key to the model’s success, however, because without it the model would result in complete separation of the data and become deprecated once the number of variables exceeds the number of observations (310).

Using the same process as above, I tried to use Principal Component Analysis with the addition of 1000 random variables. The results are even worse than with the 500 random variables. Here, it would require including approximately 200 principal components in order to capture only 80 percent of the cumulative variance in the data. Therefore, again, the best model is simply with stepwise variable selection which results in an accuracy score of 0.6774, with 210 true positives and 100 false positives. This is the result of the model predicting every observation is abnormal, and the accuracy score converges to the ratio of abnormal and normal observations (210/310 = 0.6774). Thus, this model is essentially useless and logistic regression fails completely when there are over 1000 random variables and only 310 observations. Below is a plot which summarizes the performance of logistic regression when cross-validation, forward stepwise variable selection, and accuracy score performance optimizing are all used. Note that an accuracy of 0.6774 is a result of all abnormal predictions and is therefore useless.



Moving on to the next model type, support vector machines are another very powerful and commonly used machine learning algorithm for classification. SVM’s try to separate the data by drawing support vectors from each observation to a specified area of division called a decision boundary. Since most data cannot be perfectly split by a decision boundary, the SVM’s goal is to find the decision boundary that results in the highest number of correct predictions (Ng). In two dimensions, this decision boundary is a simple line, but as the dimensions increase, the dimensions of the boundary also increase. The decision boundary that produces the best predictions for the training data is called the optimal margin classifier (Ng). In R, I am using the package e1071's svm() method to create support vector machines. As I did with logistic regression, I first model only the real data, before incrementally increasing the number of random variables to observe how support vector machines are able to find the signal in the noise.

In order to train a support vector machine, I again created a training and test sets with the same random seed that I used for my logistic regression models. The initial model is fit using the 6 real spinal measurement variables, with the provided random variables left out for later use.

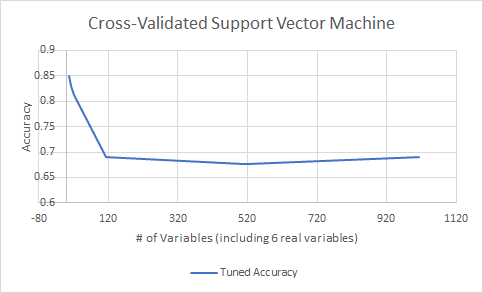
After training the model on the training set, and using the test set to make predictions, the model has an accuracy of 0.887, with 38 true positives, 5 false positives, 17 true negatives, and 2 false negatives, with positives representing abnormal classification and negatives representing normal classification. This model appears to perform very well. Using the tune.svm() function, which tunes the model parameters “gamma” and “cost” by trying variations of the parameters. The gamma parameter changes how far the influence of each training observation reaches, or how much weight is given to each training observation. The cost parameter defines how much the model should avoid misclassification during its optimization. After tuning, the accuracy score increases to 0.9032, with 38 true positives, 4 false positives, 18 true negatives, and 2 false negatives. Both of these SVM models use the train/test split method. Next, I implement cross-validation with SVM’s.

Cross-validation for support vector machines uses the same process as cross-validation for logistic regression except without the stepwise variable selection and accuracy score tuning methods. Here, the support vector machine tune.svm() function handles the tuning of the parameters and SVM’s naturally optimize the performance measures. The support vector machine with cross-validation performs quite well. After tuning, the accuracy score is 0.8484, with 186 true positives, 23 false positives, 77 true negatives, and 24 false negatives. This is much more indicative of the true performance of the initial SVM model because through cross-validation, a prediction is made for every row in the dataset, rather than only 20 percent of the data. Next, I evaluate how SVM's handle random noise by adding in the provided random variables, then incrementally adding additional random variables as I did with logistic regression.

After adding in the provided random variables and refitting the model with cross-validation, the results are an accuracy of 0.8290, with 184 true positives, 27 false positives, 73 true negatives, and 26 false negatives, with positives representing abnormal classification and negatives representing normal classification. By adding the 6 random variables, the model had both more false positives and more false negatives.

With the addition of 10 new random variables drawn from the standard normal distribution, the support vector machine's performance worsened slightly. With cross-validation, the model had an accuracy of 0.8129, 181 true positives, 29 false positives, 71 true negatives, and 29 false negatives. Next, I add 100 random variables drawn from the standard normal distribution. After training the model with cross-validation and parameter tuning, the accuracy score is 0.6903, with 169 true positives, 55 false positives, 45 true negatives, and 41 false negatives. The main concern with this model is that there are nearly as many false negatives as true negatives, indicating it does a poor job of identifying normal classifications. In terms of accuracy, this model performs only slightly better than a model which predicts all abnormal classifications.

With 500 random variables, the model breaks down completely and predicts every value to be abnormal resulting in 210 true positives, and 100 false negatives, with 0 true negatives, and 0 false positives. Thus, the accuracy score of this model is simply the ratio of abnormal to normal spines in the dataset: 0.6774. This indicates that with 500 random variables the model is essentially useless. With 1000 random variables, the model actually performs slightly better than the model with 500 random variables. The accuracy score of the model is 0.6903, with 209 true positives, 95 false positives, 5 true negatives, and 1 false positive. Although still a poor performance, it is interesting that this model with 1000 random variables performed slightly better than the model with 500 random variables. Below is a plot which summarizes the accuracy score of the tuned, cross-validated models as the number of random variables increases. The accuracy score drops below 0.70 before 120 variables, which indicates SVM’s do not handle excessive random noise very well.



Next, random forest models train multiple decision trees, and average the results to come up with a final model that is composed of elements from several decision trees. The algorithm grows each tree by selecting *m* variables at each node and the best split is used for these variables at each node. The benefit of random forest models over a single decision tree is that they tend to help reduce overfitting of the data by fitting many different trees and voting on the results of many trees (“Random Forests”). First, I train a random forest model by using the train/test split method before implementing cross validation and adding random noise. The tuneRF function is a built-in method for tuning the “mtry” parameter of the random forest model in order to optimize the predictions. The mtry parameter tells the random forest model how many variables to randomly select at each split in a tree.

The random forest model fit only on the 6 real variables and using the train/test split method has an accuracy score of 0.8065, 38 true positives, 2 false positives, 12 true negatives, and 10 false negatives. After running the tuneRF function on this model, the accuracy improved to 0.8306, with 152 true positives, 18 false positives, 53 true negatives, and 25 false negatives with positives representing abnormal classification and negatives representing normal classification. Notice that the confusion matrix for the tuned model does not add up to 310. This is because the tuneRF function produces a confusion matrix for the training set, which here is only 248 variables. Despite this weakness, the tune function does still give an idea of how much room for improvement there is in the model through the tuning of parameters. In the rest of the random forest models, I run the tune function first and then use the optimal mtry value to train the model. The importance function for the tuned random forest model indicates that the most important variable is degree\_spondylolisthesis with a MeanDecreaseGini value of 39.96. The next most important variables in the model are pelvic\_radius, sacral\_slope, and pelvic\_incidence with MeanDecreaseGini values of 16.39, 13.11, and 13.60 respectively. This MeanDecreaseGini value is the total decrease in node impurities from splitting on the variable, averaged

over all trees, measured by the Gini index.

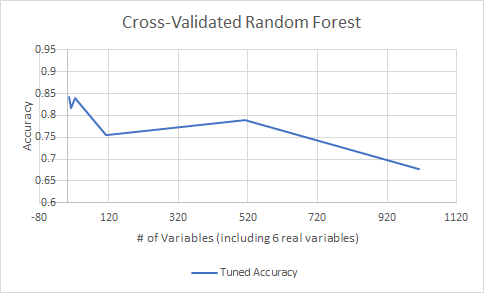
In order to implement cross validation, I use the same process as I did with the other models. The cross-validated initial model uses only the 6 real variables. After training the model five times and making predictions accordingly, the cross-validated initial model has an accuracy of 0.8419, with187 true positives, 23 false positives, 74 true negatives, and 26 false negatives, with positives representing abnormal classification and negatives representing normal classification. Here, the accuracy score increased, but in general, the accuracy score of the cross-validated model should be relatively similar to the accuracy score of the train/test split method. If the cross-validated accuracy score decreases significantly, it is likely that the train/test split method is overfitting.

The rest of the random forest models will be trained using cross-validation. Next, I train a model with the provided random noise variables added. The cross-validated and tuned random forest model using all twelve provided variables has an accuracy of 0.8161and 188 true positives, 22 false positives, 65 true negatives, and 35 false negatives, with positives representing abnormal classification and negatives representing normal classification. Considering the number of variables doubled, the model performed relatively well. For this model, the importance function indicates that the three most important variables were degree\_spondylolisthesis, pelvic\_radius, and pelvic tilt. A key element of the varImpPlots in Appendix IV is that the meaningful variables are all more important than the random variables.

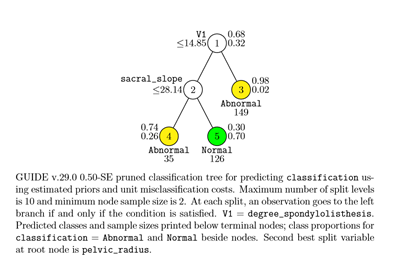
Next, I add 10 more random variables to the model consisting of randomly drawn values from a standard normal distribution. These new variables are labeled X1, X2, ... X10. The cross-validated and tuned model with the 12 provided variables plus the 10 newly created random variables results in an accuracy score of 0.8387, with 185 true positives, 25 false positives, 75 true negatives, and 25 false negatives. The importance function output indicates that the six real variables were again the most important, with the provided random variables and the newly created random variables varying in low importance. Again, degree\_spondylolisthesis is by far the most important variable.

Next, I trained the cross-validated and tuned random forest model with 100 variables drawn from the standard normal distribution in addition to the 12 provided variables. The newly added random variables are again named X1, X2, ... X100, respectively. This model produced an accuracy score of 0.7548, 192 true positives, 18 false positives, 42 true negatives, and 58 false negatives, with positives representing abnormal classification and negatives representing normal classification. This model’s main weakness is that there are more false negatives than true negatives. The importance function again shows the six real variables at the top, with the random variables scattered below.

After adding 500 random variables from the standard normal distribution to the provided 12 variables, a cross-validated and tuned random forest model is trained. The results of the model is an accuracy score of 0.7903, 182 true positives, 28 false positives, 63 true negatives, and 37 false negatives, with positives representing abnormal classification and negatives representing normal classification. Again, the variable importance indicates the six real variables are the most important, with the random variables scattered below.

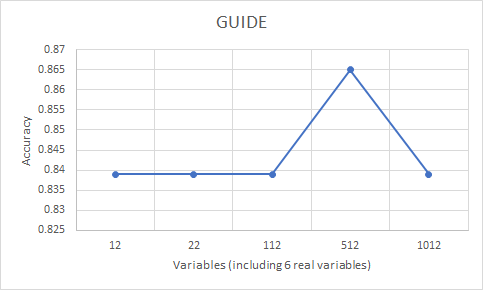
Finally, I add 1000 random variables to the 12 provided variables, again drawn from a standard normal distribution. For this model, there are a total of 1012 variables and still only 310 observations. After training the cross-validated and tuned model, the accuracy score is 0.6774, 210 true positives, 100 false positives, 0 true negatives, and 0 false negatives, with positives representing abnormal classification and negatives representing normal classification. This model is simply classifying all of the observations as abnormal; thus, the number of true positives is equal to the number of abnormal observations in the data. The plot below summarizes the performance of the tuned random forest models with respect to accuracy score.

The last model I am going to use with the spine data is GUIDE, created by Wei-Yin Loh at the University of Wisconsin, Madison. GUIDE stands for “Generalized, Unbiased, Interaction Detection and Estimation”, and the algorithm is based on decision trees. One of the options in GUIDE is to create an ensemble model similar to a random forest collection of trees. However, for this analysis, I decided to simply use the single tree model fitting methods of GUIDE. For simplicity, I opted to use all of the default settings that come with the GUIDE program, which include model fitting a single tree for classification, estimated prior probabilities, unit misclassification costs, and other default options. The GUIDE modeling process involves first creating a data description file that tell GUIDE how to handle missing values, variable types, whether or not the data includes headers, etc. Next, GUIDE creates a .IN file with all of the selected settings and type of classification. Finally, the program reads the data description file, the actual data file, and the .IN file, fits a tree to the data, and produces several files including the .OUT file which summarizes the output, a text file with the stored fitted classifications and actual classifications, and finally a .TEX file with the LaTex code for the decision tree diagram.

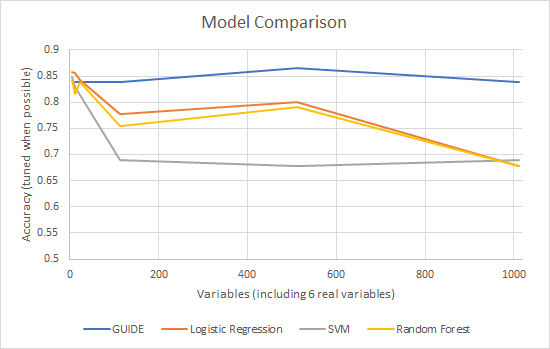
I first trained a GUIDE decision tree with just the 12 provided variables, including the six provided random variables. The results were an accuracy score of 0.839 with 172 true positives, 12 false positives, 88 true negatives, and 38 false negatives, with abnormal classification as positives, and normal classification as negatives. Each time a tree is fitted with GUIDE, a tree diagram is produced. Below is the tree diagram for the model with just the provided 12 variables. The tree is interpreted as follows: V1 is degree\_spondylolisthesis and if its value is less than or equal to 14.85 and sacral\_slope is less than 28.14, then the observation is classified as abnormal, if degree\_spondylolisthesis is less than or equal to 14.85 and sacral\_slope is greater than 28.14 then the observation is classified as normal, and if degree\_spondylolisthesis is greater than 14.85, then the observation is classified as abnormal. For each leaf node in the diagram, the proportions given indicate how many of the training observations are abnormal and normal. For example, node number three is composed of 98 percent abnormal observations and 2 percent normal observations, thus observations that classify into that leaf are classified as abnormal. The remaining trees for the other GUIDE models can be interpreted similarly and can be found in Appendix V. 

Next, I added 10 random variables drawn from the standard normal distribution and repeated the tree fitting process. The results of this GUIDE tree were the exact same as the tree fit with just the 12 provided variables and the tree diagram is in Appendix V, tree 2. Next, I added 100 random variables to the data, and the results of the model were again the same as the original model, and the tree diagram can be found in Appendix V, tree 3. The GUIDE model fit with 500 random variables did change slightly from the previous models. The model resulted in 199 true positives, 31 false positives, 69 true negatives, and 11 false negatives, with an accuracy score of 0.865. The tree diagram can be found in Appendix V, tree 4. Finally, I fit the GUIDE model with 1000 random variables. The results of this model were the same as the first 3 GUIDE models, and the tree diagram can be found in Appendix V, tree 5.

In order to calculate variable importance with the GUIDE, the program must be run again with “model importance” option selected instead of “model fitting”. The most important variables throughout the GUIDE tree models were: degree\_spondylolisthesis, pelvic\_radius, pelvic\_tilt, pelvic\_incidence, lumbar\_lordosis\_angle, and sacral\_slope, all of which are the real, meaningful spinal measurement variables. GUIDE’s main advantage is its pruning process. The algorithm quickly prunes away the variables that are not significant. Below is a plot which summarizes the accuracy score of the GUIDE models. Notice the scale on the y-axis begins at 0.825.



Overall, there is quite a large difference in performance between types of models with increasing levels of noise. GUIDE clearly performed the best, with logistic regression and random forest performing very similarly. One reason these models performed better than SVM’s is because they had more effective variable selection methods. Additionally, the plot below shows the accuracy scores, tuned when possible, vs. the number of variables in the model. The models’ performance tended to rely heavily on the tuning and variable selection process, especially when the excess random variables were introduced. Thus, the models with strong tuning and variable selection performed stronger than those that did not. Random forest and logistic regression both had a sound tuning process and an explicit variable selection procedure, resulting in rather strong performance until the excess data was overwhelming in comparison to the number of observations. Although GUIDE is somewhat of a “black box” to me, I do know that it is based on decision trees with several improvements such as unbiased variable selection and pairwise interaction detection at each node in a tree.

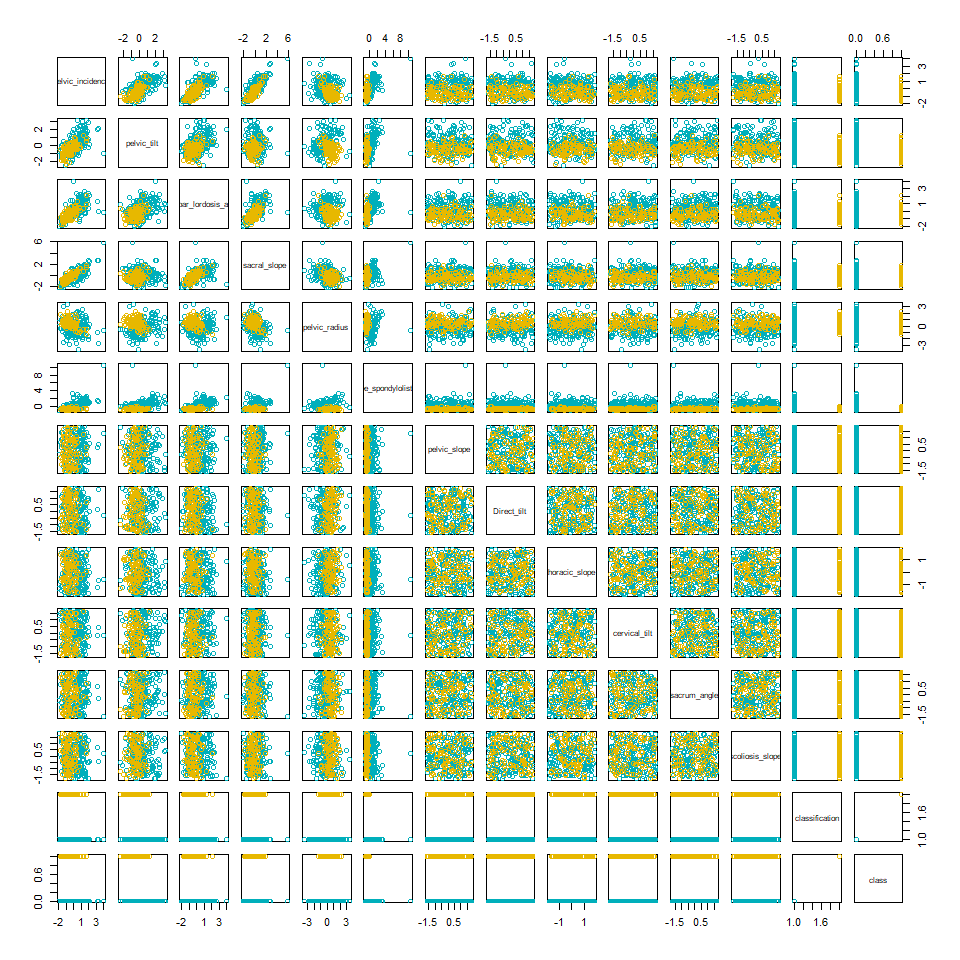


Given time, I am confident that the performance of each of these models could be improved in some way. However, the goal of this project was to compare how the selected models handled incrementally added random noise and subsequently kept track of the signal in the noise. GUIDE definitely appears to have outperformed each of the classic machine learning models, even with only its default settings and a single tree, which indicates the GUIDE program is very powerful. The process of comparing machine learning models has provided valuable insight into the importance of model tuning and variable selection processes.

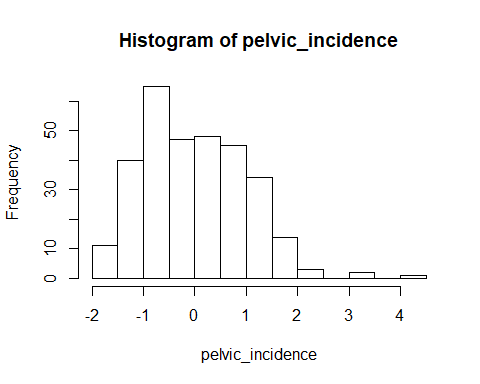
Appendix I: Data Exploration

setwd("C:/Users/Michael Streyle/Desktop/Senior Project") #laptop  
#setwd("C:/Users/Michael/Desktop/Senior Project") #desktop  
  
data1 <- read.csv('Dataset\_spine.csv', col.names = c('pelvic\_incidence', 'pelvic\_tilt', 'lumbar\_lordosis\_angle', 'sacral\_slope', 'pelvic\_radius', 'degree\_spondylolisthesis', 'pelvic\_slope', 'Direct\_tilt', 'thoracic\_slope', 'cervical\_tilt', 'sacrum\_angle', 'scoliosis\_slope', 'classification', ' '))  
  
  
data1$X. <- NULL #dropping the column with variable descriptions in it  
par (mfrow = c(1,1))  
  
  
data = scale(data1[, 1:12]) #scaling all except classification variable  
data = data.frame(data)  
data$classification = data1$classification #add classification back into scaled dataframe  
data$class = ifelse(data$classification == "Abnormal", 0, 1)

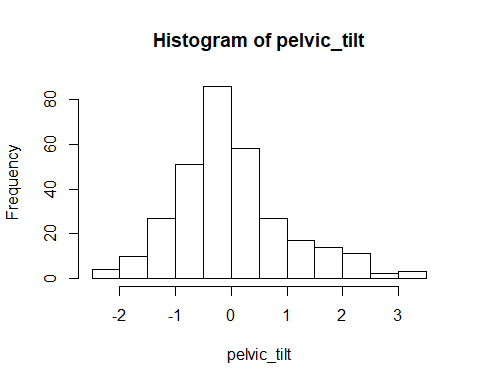
my\_cols <- c("#00AFBB", "#E7B800", "#FC4E07")  
  
pairs(data, col = my\_cols[data$classification]) #pairs plot



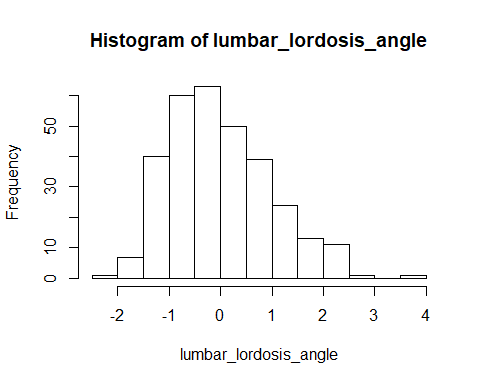
#histograms  
hist(pelvic\_incidence)



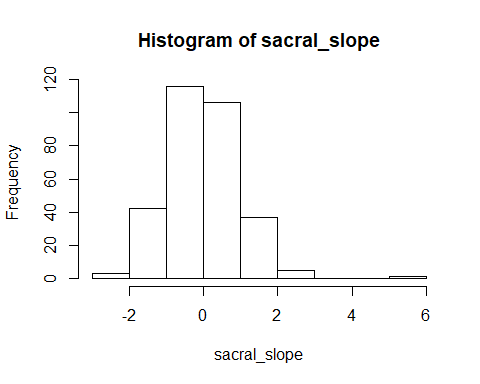
hist(pelvic\_tilt)



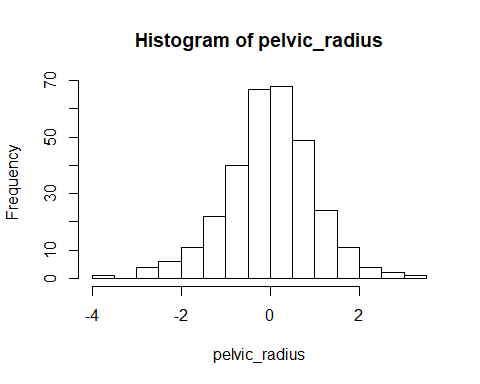
hist(lumbar\_lordosis\_angle)



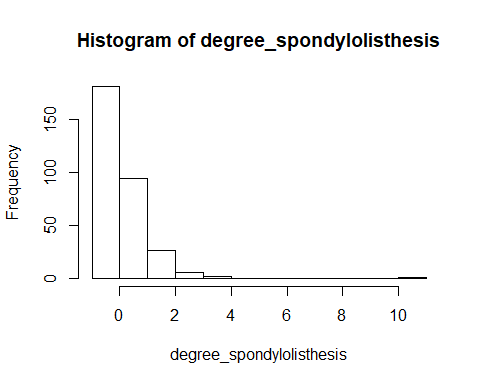
hist(sacral\_slope)



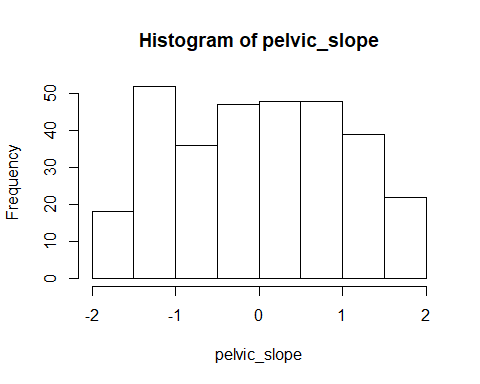
hist(pelvic\_radius)



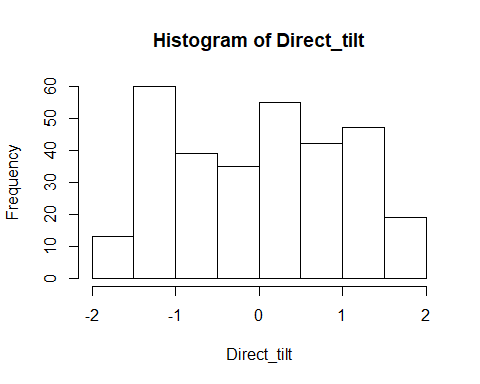
hist(degree\_spondylolisthesis)



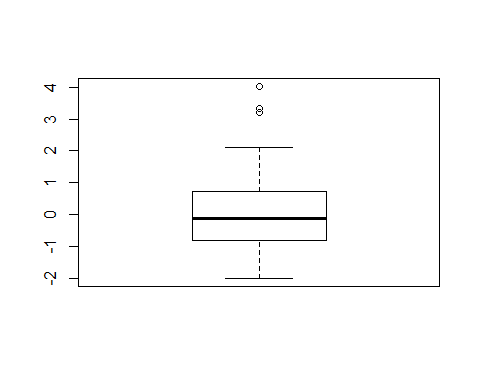
hist(pelvic\_slope)



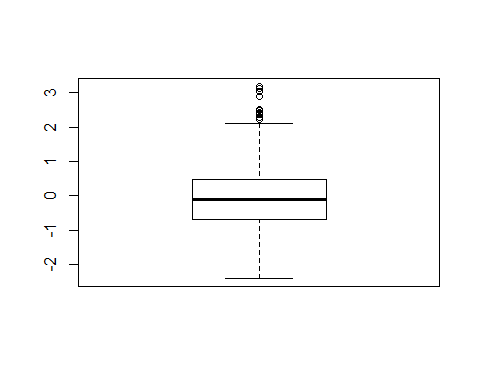
hist(Direct\_tilt)



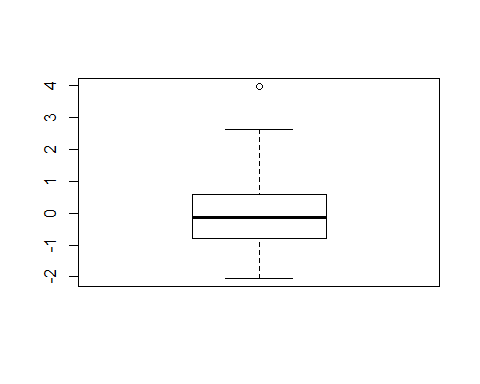
#boxplots  
boxplot(pelvic\_incidence)



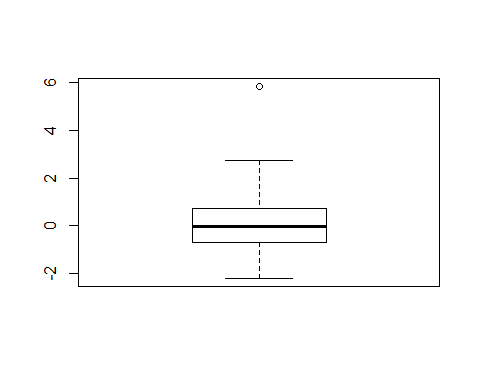
boxplot(pelvic\_tilt)



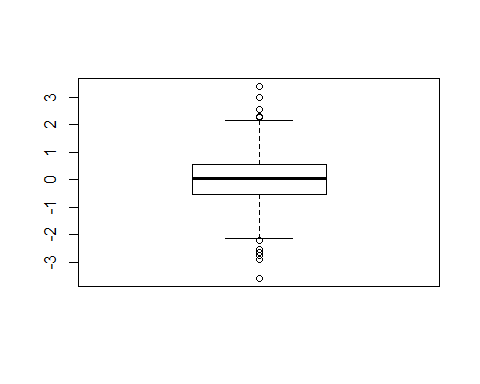
boxplot(lumbar\_lordosis\_angle)



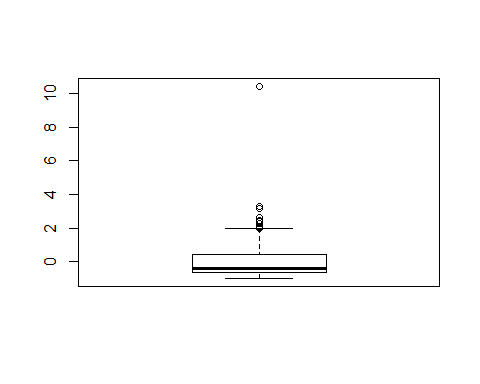
boxplot(sacral\_slope)



boxplot(pelvic\_radius)



boxplot(degree\_spondylolisthesis)



Appendix II: Logistic Regression

knitr::opts\_chunk$set(comment=NA, message=FALSE, echo=TRUE, warning = FALSE)

# Beginning

set.seed(12)  
setwd("C:/Users/Michael Streyle/Desktop/Senior Project")  
#setwd("C:/Users/Michael/Desktop/Senior Project")  
  
data1 <- read.csv('Dataset\_spine.csv', col.names = c('pelvic\_incidence', 'pelvic\_tilt', 'lumbar\_lordosis\_angle', 'sacral\_slope', 'pelvic\_radius', 'degree\_spondylolisthesis', 'pelvic\_slope', 'Direct\_tilt', 'thoracic\_slope', 'cervical\_tilt', 'sacrum\_angle', 'scoliosis\_slope', 'classification', ' '))  
  
  
data1$X. <- NULL #dropping the column with variable descriptions in it  
  
  
  
data = scale(data1[, 1:12]) #scaling all except classification variable  
data = data.frame(data)  
data$classification = data1$classification #add classification back into scaled dataframe  
data$class = ifelse(data$classification == "Abnormal", 0, 1) #making classification numeric  
  
#these are the packages I use:  
  
library(ModelMetrics) #simple confusion matrices  
library(e1071) #SVM models  
library(randomForest) #random forest models  
library(caret) #for confusion matrices longer output and factor columns  
  
attach(data)

# Initial Model

#make train and test set  
set.seed(12)  
smp\_size <- floor(0.8 \* nrow(data))  
train\_ind <- sample(seq\_len(nrow(data)), size = smp\_size)  
lr\_train <- data[train\_ind, ]  
lr\_test <- data[-train\_ind, ]  
  
  
#creating a first order model from the training data  
back.logit <- suppressWarnings(glm(classification ~ pelvic\_tilt + pelvic\_incidence + lumbar\_lordosis\_angle + sacral\_slope + pelvic\_radius + degree\_spondylolisthesis, data=lr\_train, family=binomial(link="logit")))  
  
pred = predict(back.logit, lr\_test, type = 'response')  
  
summary(back.logit)

Call:  
glm(formula = classification ~ pelvic\_tilt + pelvic\_incidence +   
 lumbar\_lordosis\_angle + sacral\_slope + pelvic\_radius + degree\_spondylolisthesis,   
 family = binomial(link = "logit"), data = lr\_train)  
  
Deviance Residuals:   
 Min 1Q Median 3Q Max   
-2.40458 -0.39229 -0.04114 0.40079 2.80457   
  
Coefficients:  
 Estimate Std. Error z value Pr(>|z|)   
(Intercept) -3.304e+00 5.267e-01 -6.274 3.52e-10 \*\*\*  
pelvic\_tilt -5.314e+08 4.441e+08 -1.197 0.23146   
pelvic\_incidence 9.151e+08 7.648e+08 1.197 0.23146   
lumbar\_lordosis\_angle 9.587e-02 4.788e-01 0.200 0.84132   
sacral\_slope -7.127e+08 5.956e+08 -1.197 0.23146   
pelvic\_radius 1.147e+00 3.156e-01 3.633 0.00028 \*\*\*  
degree\_spondylolisthesis -6.375e+00 9.626e-01 -6.623 3.52e-11 \*\*\*  
---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
  
(Dispersion parameter for binomial family taken to be 1)  
  
 Null deviance: 308.84 on 247 degrees of freedom  
Residual deviance: 141.83 on 241 degrees of freedom  
AIC: 155.83  
  
Number of Fisher Scoring iterations: 7

#model with interaction  
logit\_int = suppressWarnings(glm(classification ~ pelvic\_tilt + pelvic\_incidence + lumbar\_lordosis\_angle + sacral\_slope + pelvic\_radius + degree\_spondylolisthesis + degree\_spondylolisthesis:pelvic\_radius, data=lr\_train, family=binomial(link="logit")))  
#summary(logit\_int)  
int\_pred = predict(logit\_int, lr\_test, type= "response")  
  
  
min.model = suppressWarnings(glm(class ~ 1, data=lr\_train, family=binomial(link="logit")))  
biggest <- formula(glm(class ~ . - classification,lr\_train, family=binomial(link="logit")))  
  
step.logit = step(min.model, direction='forward', scope=biggest, trace = 0)  
step\_pred = predict(object = step.logit, lr\_test, type='response')  
  
summary(step.logit)

Call:  
glm(formula = class ~ degree\_spondylolisthesis + sacral\_slope +   
 pelvic\_radius + pelvic\_tilt + Direct\_tilt, family = binomial(link = "logit"),   
 data = lr\_train)  
  
Deviance Residuals:   
 Min 1Q Median 3Q Max   
-2.07271 -0.37358 -0.04328 0.40737 2.63670   
  
Coefficients:  
 Estimate Std. Error z value Pr(>|z|)   
(Intercept) -3.3643 0.5325 -6.318 2.64e-10 \*\*\*  
degree\_spondylolisthesis -6.3018 0.9480 -6.648 2.98e-11 \*\*\*  
sacral\_slope 1.5558 0.3547 4.386 1.15e-05 \*\*\*  
pelvic\_radius 1.1881 0.3089 3.846 0.00012 \*\*\*  
pelvic\_tilt -0.7932 0.3237 -2.451 0.01426 \*   
Direct\_tilt -0.3782 0.2264 -1.671 0.09473 .   
---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
  
(Dispersion parameter for binomial family taken to be 1)  
  
 Null deviance: 308.84 on 247 degrees of freedom  
Residual deviance: 140.48 on 242 degrees of freedom  
AIC: 152.48  
  
Number of Fisher Scoring iterations: 7

#interaction plot between dgree\_spondylolisthesis and pelvic\_radius  
#interaction.plot(x.factor = pelvic\_radius, trace.factor = degree\_spondylolisthesis, response = classification)  
  
#odds ratios  
#expbetas = exp(back.logit$coefficients)  
#expbetas  
  
  
#confusion matrix for logistic regression without interaction  
lr\_test$lr\_probs = pred  
lr\_cfr = ModelMetrics::confusionMatrix( predicted = lr\_test$lr\_probs, actual = lr\_test$class)   
lr\_cfr

[,1] [,2]  
[1,] 36 6  
[2,] 4 16

#confusion matrix for logistic regression with interaction  
int\_cf = ModelMetrics::confusionMatrix( predicted = int\_pred, actual = lr\_test$class)  
int\_cf

[,1] [,2]  
[1,] 36 7  
[2,] 4 15

#accuracy score for Logistic Regression without interaction  
acc\_lr = (lr\_cfr[1,1] + lr\_cfr[2,2])/nrow(lr\_test)  
acc\_lr

[1] 0.8387097

#accuracy score for LR with interaction  
acc\_lr\_int = (int\_cf[1,1] + int\_cf[2,2])/nrow(lr\_test)  
acc\_lr\_int

[1] 0.8225806

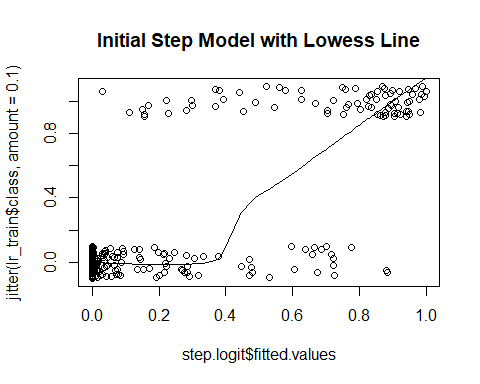
#confusion matrix for logistic regression with forward stepwise  
lr\_test$lr\_probs\_step = step\_pred  
lr\_cfr\_step = ModelMetrics::confusionMatrix( predicted = lr\_test$lr\_probs\_step, actual = lr\_test$class)   
lr\_cfr\_step

[,1] [,2]  
[1,] 37 6  
[2,] 3 16

#accuracy score for forward stepwise model  
acc\_lr\_step = (lr\_cfr\_step[1,1] + lr\_cfr\_step[2,2])/nrow(lr\_test)  
acc\_lr\_step

[1] 0.8548387

plot(x = step.logit$fitted.values, y= jitter(lr\_train$class, amount = .1), main = "Initial Step Model with Lowess Line")  
lines(lowess(x = step.logit$fitted.values, y= jitter(lr\_train$class, amount = .1)))



# Find Optimal Cutoff for Train/Test Initial Model with Forward Stepwise

set.seed(12)  
cutoffs = c()  
accuracies = c()  
train\_pred = predict(step.logit, lr\_train, type='response')  
probs = as.vector(train\_pred)#attr(train\_pred, "probabilities")  
for (val in probs){  
 cf = ModelMetrics::confusionMatrix(lr\_train$class, probs, cutoff = val)  
 acc = (cf[1,1] + cf[2,2])/nrow(lr\_train)  
 accuracies = c(accuracies, acc)  
 cutoffs = c(cutoffs, val)  
}  
  
max1 = max(accuracies)  
ind = match(max1, accuracies)  
op = cutoffs[ind]  
  
test\_pred = predict(step.logit, lr\_test, type='response')  
  
op\_lr = ModelMetrics::confusionMatrix(actual = lr\_test$class, predicted = test\_pred, cutoff = op)  
op\_lr

[,1] [,2]  
[1,] 40 16  
[2,] 0 6

op\_acc\_lr = (op\_lr[1,1] + op\_lr[2,2])/nrow(lr\_test)  
op\_acc\_lr

[1] 0.7419355

# Cross-Validation with Initial Model with Optimizing with Set Seed

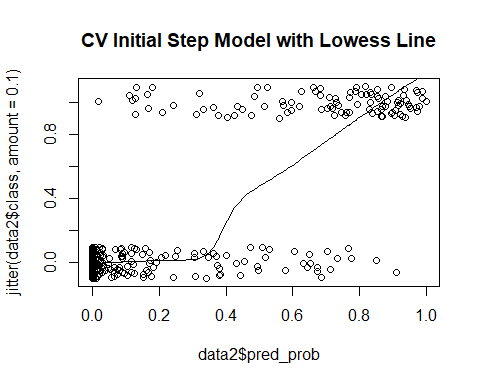
set.seed(12)  
data$rand\_int = runif(n=nrow(data), min = 0, max = 1) #random values from uniform distribution  
data2 = data[order(data$rand\_int),] #reorder dataframe by random value  
  
data2$group = seq(from = 1, to=5, by=1)  
#detach(data2)  
data2$pred\_prob = 0  
attach(data2)  
  
#Logistic Regression Cross-Validation  
data2$pred\_cv = 0  
  
for (grp in 1:5){  
 train = data2[data2$group != grp, ]  
 test = data2[data2$group == grp,]  
   
 cv\_min.model = suppressWarnings(glm(class ~ 1, data=train, family=binomial(link="logit")))  
   
 biggest <- suppressWarnings(formula(glm(class ~ pelvic\_tilt + pelvic\_incidence + lumbar\_lordosis\_angle + sacral\_slope + pelvic\_radius + degree\_spondylolisthesis, train, family=binomial(link="logit"))))  
  
 step.logit = step(min.model, direction='forward', scope=biggest, trace=0)  
 pred = predict(step.logit, train, type = 'response')  
 cutoffs = c()  
 accuracies = c()  
 for (val in as.vector(pred)){  
 cf = ModelMetrics::confusionMatrix(train$class, as.vector(pred), cutoff = val)  
 acc = (cf[1,1] + cf[2,2])/nrow(train)  
 accuracies = c(accuracies, acc)  
 cutoffs = c(cutoffs, val)  
 }  
  
 max1 = max(accuracies)  
 ind = match(max1, accuracies)  
 op = cutoffs[ind]  
   
 cv\_lr\_pred = predict(step.logit, test, type = 'response')  
 data2$pred\_prob[data2$group == grp] = cv\_lr\_pred  
 data2$pred\_cv[data2$group == grp] = ifelse(cv\_lr\_pred < op, 0, 1)  
}  
  
cv\_lr\_cfr = ModelMetrics::confusionMatrix( predicted = data2$pred\_cv, actual = data2$class)  
cv\_lr\_cfr

[,1] [,2]  
[1,] 189 23  
[2,] 21 77

#accuracy score for Cross Validated Logistic Regression  
acc\_cv\_lr = (cv\_lr\_cfr[1,1] + cv\_lr\_cfr[2,2])/nrow(data2)  
acc\_cv\_lr

[1] 0.8580645

plot(x = data2$pred\_prob, y= jitter(data2$class, amount = .1), main = "CV Initial Step Model with Lowess Line")  
lines(lowess(x = data2$pred\_prob, y= jitter(data2$class, amount = .1)))



# Accuracy Distribution of Cross-Validation with Initial Model with Optimizing

# dist\_acc = c()  
#   
# for (n in 1:100){  
# data$rand\_int = runif(n=nrow(data), min = 0, max = 1) #random values from uniform distribution  
# data2 = data[order(data$rand\_int),] #reorder dataframe by random value  
#   
# data2$group = seq(from = 1, to=5, by=1)  
#   
# attach(data2)  
#   
# #Logistic Regression Cross-Validation  
# data2$pred\_cv = 0  
#   
# for (grp in 1:5){  
# train = data2[data2$group != grp, ]  
# test = data2[data2$group == grp,]  
#   
# cv\_min.model = glm(class ~ 1, data=train, family=binomial(link="logit"))  
#   
# biggest <- formula(glm(class ~ pelvic\_tilt + pelvic\_incidence + lumbar\_lordosis\_angle + sacral\_slope + pelvic\_radius + degree\_spondylolisthesis, train, family=binomial(link="logit")))  
#   
# step.logit = step(min.model, direction='forward', scope=biggest)  
# pred = predict(step.logit, train, type = 'response')  
# cutoffs = c()  
# accuracies = c()  
# for (val in as.vector(pred)){  
# cf = ModelMetrics::confusionMatrix(train$class, as.vector(pred), cutoff = val)  
# acc = (cf[1,1] + cf[2,2])/nrow(train)  
# accuracies = c(accuracies, acc)  
# cutoffs = c(cutoffs, val)  
# }  
#   
# max1 = max(accuracies)  
# ind = match(max1, accuracies)  
# op = cutoffs[ind]  
#   
# cv\_lr\_pred = predict(step.logit, test, type = 'response')  
#   
# data2$pred\_cv[data2$group == grp] = ifelse(cv\_lr\_pred < op, 0, 1)  
# }  
#   
# cv\_lr\_cfr = ModelMetrics::confusionMatrix( predicted = data2$pred\_cv, actual = data2$class)  
# cv\_lr\_cfr  
#   
# #accuracy score for Cross Validated Logistic Regression  
# acc\_cv\_lr = (cv\_lr\_cfr[1,1] + cv\_lr\_cfr[2,2])/nrow(data2)  
# acc\_cv\_lr  
#   
# dist\_acc = c(dist\_acc, acc\_cv\_lr)  
# }  
#   
#   
# hist(dist\_acc)

# Add Provided Random Noise

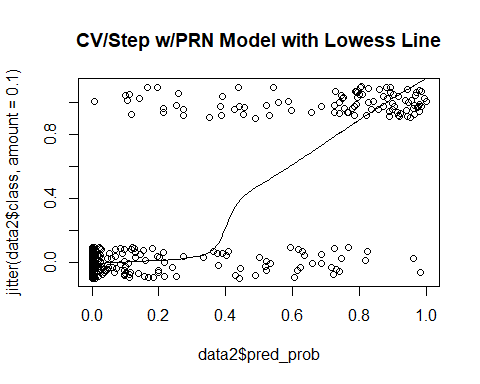
set.seed(12)  
df <- read.csv('Dataset\_spine.csv', col.names = c('pelvic\_incidence', 'pelvic\_tilt', 'lumbar\_lordosis\_angle', 'sacral\_slope', 'pelvic\_radius', 'degree\_spondylolisthesis', 'pelvic\_slope', 'Direct\_tilt', 'thoracic\_slope', 'cervical\_tilt', 'sacrum\_angle', 'scoliosis\_slope', 'classification', ' '))  
  
df$class = ifelse(df$classification == "Abnormal", 0, 1) #making classification numeric  
df$X. <- NULL #dropping the column with variable descriptions in it  
  
  
  
data = scale(df[, 1:12]) #scaling all except classification variable  
data = data.frame(data)  
data$class = df$class #add classification back into scaled dataframe  
data$classification = df$classification  
  
data$rand\_int = runif(n=nrow(data), min = 0, max = 1) #random values from uniform distribution  
data2 = data[order(data$rand\_int),] #reorder dataframe by random value  
  
data2$group = seq(from = 1, to=5, by=1)  
data2$pred\_cv = 0  
detach(data2)  
data2$pred\_prob = 0  
attach(data2)  
  
  
for (grp in 1:5){  
 train = data2[data2$group != grp, ]  
 test = data2[data2$group == grp,]  
   
 min.model = suppressWarnings(glm(class ~ 1, data=train, family=binomial(link="logit")))  
 biggest <- suppressWarnings(formula(glm(class ~ . - classification - pred\_prob - pred\_cv - rand\_int - group, data = train, family=binomial(link="logit"))))  
  
 step.logit = step(min.model, direction='forward', scope=biggest, trace=0)  
 pred = predict(step.logit, train, type = 'response')  
 cutoffs = c()  
 accuracies = c()  
 for (val in as.vector(pred)){  
 cf = ModelMetrics::confusionMatrix(train$class, as.vector(pred), cutoff = val)  
 acc = (cf[1,1] + cf[2,2])/nrow(train)  
 accuracies = c(accuracies, acc)  
 cutoffs = c(cutoffs, val)  
 }  
  
 max1 = max(accuracies)  
 ind = match(max1, accuracies)  
 op = cutoffs[ind]  
   
 cv\_lr\_pred = predict(step.logit, test, type = 'response')  
 data2$pred\_prob[data2$group == grp] = cv\_lr\_pred  
 data2$pred\_cv[data2$group == grp] = ifelse(cv\_lr\_pred < op, 0, 1)  
}  
  
  
cv\_lr\_cfr\_wprn = ModelMetrics::confusionMatrix( predicted = data2$pred\_cv, actual = data2$class)  
cv\_lr\_cfr\_wprn #with provided random noise

[,1] [,2]  
[1,] 193 28  
[2,] 17 72

#accuracy score for Cross Validated RandomForest with Provided Random Noise  
acc\_cv\_lr\_wprn = (cv\_lr\_cfr\_wprn[1,1] + cv\_lr\_cfr\_wprn[2,2])/nrow(data2)  
acc\_cv\_lr\_wprn

[1] 0.8548387

plot(x = data2$pred\_prob, y= jitter(data2$class, amount = .1), main = "CV/Step w/PRN Model with Lowess Line")  
lines(lowess(x = data2$pred\_prob, y= jitter(data2$class, amount = .1)))



# Add 10 Random Variables

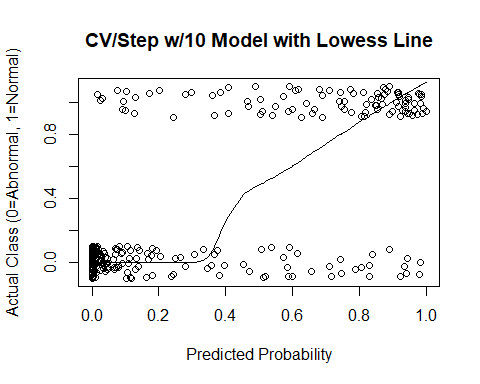
set.seed(12)  
df <- read.csv('Dataset\_spine.csv', col.names = c('pelvic\_incidence', 'pelvic\_tilt', 'lumbar\_lordosis\_angle', 'sacral\_slope', 'pelvic\_radius', 'degree\_spondylolisthesis', 'pelvic\_slope', 'Direct\_tilt', 'thoracic\_slope', 'cervical\_tilt', 'sacrum\_angle', 'scoliosis\_slope', 'classification', ' '))  
  
df$class = ifelse(df$classification == "Abnormal", 0, 1) #making classification numeric  
df$X. <- NULL #dropping the column with variable descriptions in it  
data = scale(df[, 1:12]) #scaling all except classification variable  
data = data.frame(data)  
data$classification = df$classification #add classification back into scaled dataframe  
data$class = df$class  
rand\_df = data.frame(matrix(rnorm(10\*nrow(data)), nrow = nrow(data), ncol = 10))  
data = cbind(data, rand\_df)  
  
  
data$rand\_int = runif(n=nrow(data), min = 0, max = 1) #random values from uniform distribution  
data2 = data[order(data$rand\_int),] #reorder dataframe by random value  
  
data2$group = seq(from = 1, to=5, by=1)  
data2$pred\_cv = 0  
detach(data2)  
data2$pred\_prob = 0  
attach(data2)  
  
  
for (grp in 1:5){  
 train = data2[data2$group != grp, ]  
 test = data2[data2$group == grp,]  
   
 min.model = suppressWarnings(glm(class ~ 1, data=train, family=binomial(link="logit")))  
 biggest <- suppressWarnings(formula(glm(class ~ . - classification - pred\_cv - pred\_prob - rand\_int - group, data = train, family=binomial(link="logit"))))  
  
 step.logit = step(min.model, direction='forward', scope=biggest, trace=0)  
 pred = predict(step.logit, train, type = 'response')  
 cutoffs = c()  
 accuracies = c()  
 for (val in as.vector(pred)){  
 cf = ModelMetrics::confusionMatrix(train$class, as.vector(pred), cutoff = val)  
 acc = (cf[1,1] + cf[2,2])/nrow(train)  
 accuracies = c(accuracies, acc)  
 cutoffs = c(cutoffs, val)  
 }  
  
 max1 = max(accuracies)  
 ind = match(max1, accuracies)  
 op = cutoffs[ind]  
   
 cv\_lr\_pred = predict(step.logit, test, type = 'response')  
 data2$pred\_prob[data2$group == grp] = cv\_lr\_pred  
 data2$pred\_cv[data2$group == grp] = ifelse(cv\_lr\_pred < op, 0, 1)  
}  
  
  
  
cv\_lr\_cfr\_w10 = ModelMetrics::confusionMatrix( predicted = data2$pred\_cv, actual = data2$class)  
cv\_lr\_cfr\_w10

[,1] [,2]  
[1,] 187 26  
[2,] 23 74

#accuracy score for Cross Validated RandomForest with 10 additional random variables  
acc\_cv\_lr\_w10 = (cv\_lr\_cfr\_w10[1,1] + cv\_lr\_cfr\_w10[2,2])/nrow(data2)  
acc\_cv\_lr\_w10

[1] 0.8419355

plot(x = data2$pred\_prob, y= jitter(data2$class, amount = .1), main = "CV/Step w/10 Model with Lowess Line", xlab="Predicted Probability", ylab='Actual Class (0=Abnormal, 1=Normal)')  
lines(lowess(x = data2$pred\_prob, y= jitter(data2$class, amount = .1)))



# Add 100 Random Variables with set seed

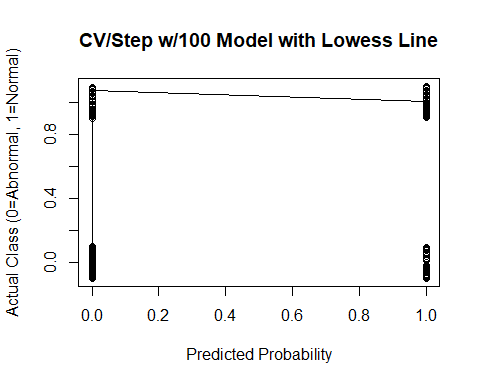
set.seed(12)  
df <- read.csv('Dataset\_spine.csv', col.names = c('pelvic\_incidence', 'pelvic\_tilt', 'lumbar\_lordosis\_angle', 'sacral\_slope', 'pelvic\_radius', 'degree\_spondylolisthesis', 'pelvic\_slope', 'Direct\_tilt', 'thoracic\_slope', 'cervical\_tilt', 'sacrum\_angle', 'scoliosis\_slope', 'classification', ' '))  
  
df$class = ifelse(df$classification == "Abnormal", 0, 1) #making classification numeric  
df$X. <- NULL #dropping the column with variable descriptions in it  
data = scale(df[, 1:12]) #scaling all except classification variable  
data = data.frame(data)  
data$classification = df$classification #add classification back into scaled dataframe  
data$class = df$class  
rand\_df = data.frame(matrix(rnorm(100\*nrow(data)), nrow = nrow(data), ncol = 100))  
data = cbind(data, rand\_df)  
  
  
data$rand\_int = runif(n=nrow(data), min = 0, max = 1) #random values from uniform distribution  
data2 = data[order(data$rand\_int),] #reorder dataframe by random value  
  
data2$group = seq(from = 1, to=5, by=1)  
data2$pred\_cv = 0  
  
detach(data2)  
data2$pred\_prob = 0  
attach(data2)  
  
  
  
for (grp in 1:5){  
 train = data2[data2$group != grp, ]  
 test = data2[data2$group == grp,]  
   
 min.model = suppressWarnings(glm(class ~ 1, data=train, family=binomial(link="logit")))  
 biggest <- suppressWarnings(formula(glm(class ~ . - classification - pred\_prob - pred\_cv - rand\_int - group, data = train, family=binomial(link="logit"))))  
  
 step.logit = step(min.model, direction='forward', scope=biggest, trace=0)  
 pred = predict(step.logit, train, type = 'response')  
 cutoffs = c()  
 accuracies = c()  
 for (val in as.vector(pred)){  
 cf = ModelMetrics::confusionMatrix(train$class, as.vector(pred), cutoff = val)  
 acc = (cf[1,1] + cf[2,2])/nrow(train)  
 accuracies = c(accuracies, acc)  
 cutoffs = c(cutoffs, val)  
 }  
  
 max1 = max(accuracies)  
 ind = match(max1, accuracies)  
 op = cutoffs[ind]  
   
 cv\_lr\_pred = predict(step.logit, test, type = 'response')  
 data2$pred\_prob[data2$group == grp] = cv\_lr\_pred  
 data2$pred\_cv[data2$group == grp] = ifelse(cv\_lr\_pred < op, 0, 1)  
}  
  
  
  
cv\_lr\_cfr\_w100 = ModelMetrics::confusionMatrix( predicted = data2$pred\_cv, actual = data2$class)  
cv\_lr\_cfr\_w100 #with provided random noise

[,1] [,2]  
[1,] 175 34  
[2,] 35 66

#accuracy score for Cross Validated RandomForest with Provided Random Noise  
acc\_cv\_lr\_w100 = (cv\_lr\_cfr\_w100[1,1] + cv\_lr\_cfr\_w100[2,2])/nrow(data2)  
acc\_cv\_lr\_w100

[1] 0.7774194

plot(x = data2$pred\_prob, y= jitter(data2$class, amount = .1), main = "CV/Step w/100 Model with Lowess Line", xlab="Predicted Probability", ylab="Actual Class (0=Abnormal, 1=Normal)")  
lines(lowess(x = data2$pred\_prob, y= jitter(data2$class, amount = .1)))



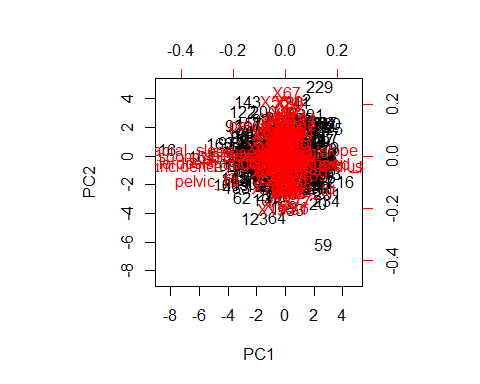
#PCA for 100 random variables  
myvars <- names(data2) %in% c("classification","class", "rand\_int", "pred\_cv", "group") #remove classificaiton  
newdata = data2[!myvars]  
  
prin\_comp = prcomp(newdata, scale.= T)  
names(prin\_comp)

[1] "sdev" "rotation" "center" "scale" "x"

prin\_comp$rotation[1:5,1:4]

PC1 PC2 PC3 PC4  
pelvic\_incidence -0.4585396 -0.03986544 0.02559957 -0.04130906  
pelvic\_tilt -0.2988447 -0.09654887 0.10297940 0.09719343  
lumbar\_lordosis\_angle -0.3962508 -0.02873156 -0.03282033 0.01412432  
sacral\_slope -0.3659877 0.02079636 -0.04390970 -0.12551259  
pelvic\_radius 0.1216077 -0.04026772 -0.04059324 0.06597922

biplot(prin\_comp, scale = 0)



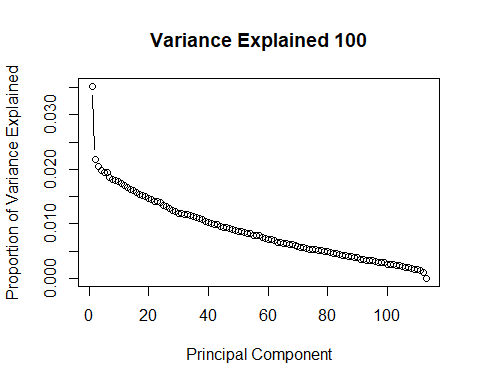
std\_dev <- prin\_comp$sdev  
pr\_var <- std\_dev^2  
pr\_var[1:10]

[1] 3.980433 2.465193 2.320331 2.234025 2.204784 2.197591 2.101477  
 [8] 2.045163 2.019319 1.999980

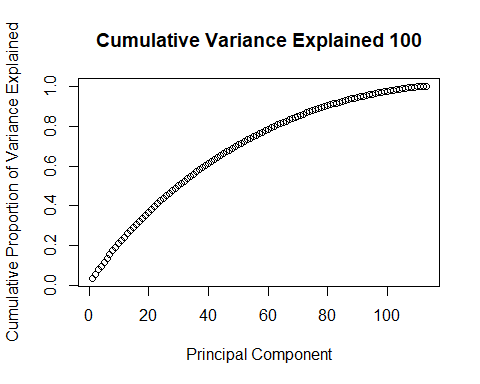
prop\_varex <- pr\_var/sum(pr\_var)  
prop\_varex[1:20]

[1] 0.03522507 0.02181587 0.02053390 0.01977013 0.01951136 0.01944771  
 [7] 0.01859714 0.01809879 0.01787008 0.01769893 0.01732758 0.01708304  
[13] 0.01661430 0.01631056 0.01610876 0.01578239 0.01542219 0.01515624  
[19] 0.01498812 0.01468698

plot(prop\_varex, xlab = "Principal Component",  
 ylab = "Proportion of Variance Explained",  
 type = "b", main='Variance Explained 100')



plot(cumsum(prop\_varex), xlab = "Principal Component",  
 ylab = "Cumulative Proportion of Variance Explained",  
 type = "b", main= "Cumulative Variance Explained 100")



# Accuracy Distribution for 100 Random Variables

# acc\_100\_dist = c()  
#   
# for (n in 1:100){  
# df <- read.csv('Dataset\_spine.csv', col.names = c('pelvic\_incidence', 'pelvic\_tilt', 'lumbar\_lordosis\_angle', 'sacral\_slope', 'pelvic\_radius', 'degree\_spondylolisthesis', 'pelvic\_slope', 'Direct\_tilt', 'thoracic\_slope', 'cervical\_tilt', 'sacrum\_angle', 'scoliosis\_slope', 'classification', ' '))  
#   
# df$class = ifelse(df$classification == "Abnormal", 0, 1) #making classification numeric  
# df$X. <- NULL #dropping the column with variable descriptions in it  
# data = scale(df[, 1:12]) #scaling all except classification variable  
# data = data.frame(data)  
# data$classification = df$classification #add classification back into scaled dataframe  
# data$class = df$class  
# rand\_df = data.frame(matrix(rnorm(100\*nrow(data)), nrow = nrow(data), ncol = 100))  
# data = cbind(data, rand\_df)  
#   
#   
# data$rand\_int = runif(n=nrow(data), min = 0, max = 1) #random values from uniform distribution  
# data2 = data[order(data$rand\_int),] #reorder dataframe by random value  
#   
# data2$group = seq(from = 1, to=5, by=1)  
# data2$pred\_cv = 0  
# attach(data2)  
#   
#   
#   
# for (grp in 1:5){  
# train = data2[data2$group != grp, ]  
# test = data2[data2$group == grp,]  
#   
# min.model = glm(class ~ 1, data=train, family=binomial(link="logit"))  
# biggest <- formula(glm(class ~ . - classification - pred\_cv - rand\_int - group, data = train, family=binomial(link="logit")))  
#   
# step.logit = step(min.model, direction='forward', scope=biggest)  
# pred = predict(step.logit, train, type = 'response')  
# cutoffs = c()  
# accuracies = c()  
# for (val in as.vector(pred)){  
# cf = ModelMetrics::confusionMatrix(train$class, as.vector(pred), cutoff = val)  
# acc = (cf[1,1] + cf[2,2])/nrow(train)  
# accuracies = c(accuracies, acc)  
# cutoffs = c(cutoffs, val)  
# }  
#   
# max1 = max(accuracies)  
# ind = match(max1, accuracies)  
# op = cutoffs[ind]  
#   
# cv\_lr\_pred = predict(step.logit, test, type = 'response')  
#   
# data2$pred\_cv[data2$group == grp] = ifelse(cv\_lr\_pred < op, 0, 1)  
# }  
#   
#   
#   
# cv\_lr\_cfr\_w100 = ModelMetrics::confusionMatrix( predicted = data2$pred\_cv, actual = data2$class)  
# cv\_lr\_cfr\_w100 #with provided random noise  
#   
# #accuracy score for Cross Validated RandomForest with Provided Random Noise  
# acc\_cv\_lr\_w100 = (cv\_lr\_cfr\_w100[1,1] + cv\_lr\_cfr\_w100[2,2])/nrow(data2)  
# acc\_cv\_lr\_w100  
# acc\_100\_dist = c(acc\_100\_dist, acc\_cv\_lr\_w100)  
# }  
#   
#   
# hist(acc\_100\_dist)

# Add 500 Random Variables

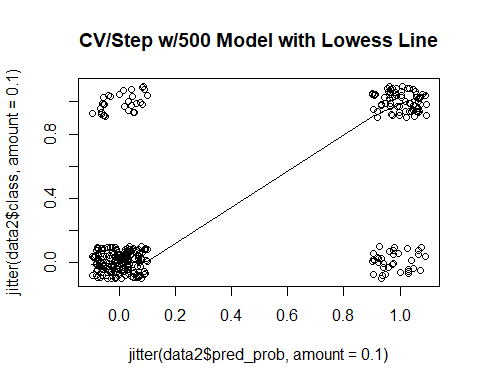
set.seed(12)  
df <- read.csv('Dataset\_spine.csv', col.names = c('pelvic\_incidence', 'pelvic\_tilt', 'lumbar\_lordosis\_angle', 'sacral\_slope', 'pelvic\_radius', 'degree\_spondylolisthesis', 'pelvic\_slope', 'Direct\_tilt', 'thoracic\_slope', 'cervical\_tilt', 'sacrum\_angle', 'scoliosis\_slope', 'classification', ' '))  
  
df$class = ifelse(df$class == "Abnormal", 0, 1) #making classification numeric  
df$X. <- NULL #dropping the column with variable descriptions in it  
data = scale(df[, 1:12]) #scaling all except classification variable  
data = data.frame(data)  
data$classification = df$classification #add classification back into scaled dataframe  
data$class = df$class  
rand\_df = data.frame(matrix(rnorm(500\*nrow(data)), nrow = nrow(data), ncol = 500))  
data = cbind(data, rand\_df)  
  
  
data$rand\_int = runif(n=nrow(data), min = 0, max = 1) #random values from uniform distribution  
data2 = data[order(data$rand\_int),] #reorder dataframe by random value  
data2$group = seq(from = 1, to=5, by=1)  
data2$pred\_cv = 0  
detach(data2)  
data2$pred\_prob = 0  
attach(data2)  
  
  
  
  
for (grp in 1:5){  
 train = data2[data2$group != grp, ]  
 test = data2[data2$group == grp,]  
   
 min.model = suppressWarnings(glm(class ~ 1, data=train, family=binomial(link="logit")))  
 biggest <- suppressWarnings(formula(glm(class ~ . - classification - pred\_cv - rand\_int - group, data = train, family=binomial(link="logit"))))  
  
 step.logit = step(min.model, direction='forward', scope=biggest, trace=0)  
 pred = predict(step.logit, train, type = 'response')  
 cutoffs = c()  
 accuracies = c()  
 for (val in as.vector(pred)){  
 cf = ModelMetrics::confusionMatrix(train$class, as.vector(pred), cutoff = val)  
 acc = (cf[1,1] + cf[2,2])/nrow(train)  
 accuracies = c(accuracies, acc)  
 cutoffs = c(cutoffs, val)  
 }  
  
 max1 = max(accuracies)  
 ind = match(max1, accuracies)  
 op = cutoffs[ind]  
   
 cv\_lr\_pred = predict(step.logit, test, type = 'response')  
 data2$pred\_prob[data2$group == grp] = cv\_lr\_pred  
 data2$pred\_cv[data2$group == grp] = ifelse(cv\_lr\_pred < op, 0, 1)  
}  
  
  
cv\_lr\_cfr\_w500 = ModelMetrics::confusionMatrix( predicted = data2$pred\_cv, actual = data2$class)  
cv\_lr\_cfr\_w500 #with provided random noise

[,1] [,2]  
[1,] 175 27  
[2,] 35 73

#accuracy score for Cross Validated RandomForest with 500 Random Variables  
acc\_cv\_lr\_w500 = (cv\_lr\_cfr\_w500[1,1] + cv\_lr\_cfr\_w500[2,2])/nrow(data2)  
acc\_cv\_lr\_w500

[1] 0.8

plot(x = jitter(data2$pred\_prob, amount = .1), y= jitter(data2$class, amount = .1), main = "CV/Step w/500 Model with Lowess Line")  
lines(lowess(x = jitter(data2$pred\_prob, amount = .1), y= jitter(data2$class, amount = .1)))



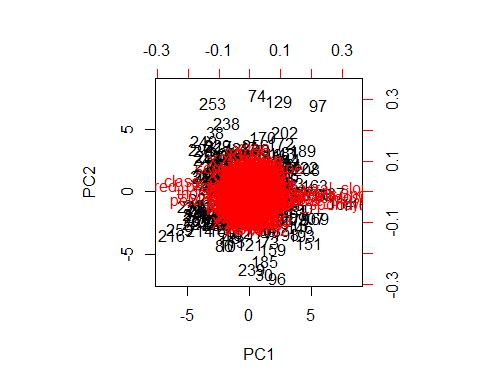
#PCA for 500 random variables  
myvars <- names(data2) %in% c("classification", "rand\_int", "pred\_cv", "group") #remove classificaiton  
newdata = data2[!myvars]  
  
prin\_comp = prcomp(newdata, scale.= T)  
names(prin\_comp)

[1] "sdev" "rotation" "center" "scale" "x"

prin\_comp$rotation[1:5,1:4]

PC1 PC2 PC3 PC4  
pelvic\_incidence 0.3427986 -0.008366553 0.025014206 0.030629291  
pelvic\_tilt 0.2150461 -0.029647281 -0.026541067 0.017892787  
lumbar\_lordosis\_angle 0.2936539 -0.017941039 -0.003384455 0.030592587  
sacral\_slope 0.2798461 0.011361719 0.051909732 0.025989929  
pelvic\_radius -0.1085754 -0.028813317 0.027042366 -0.001630254

biplot(prin\_comp, scale = 0)



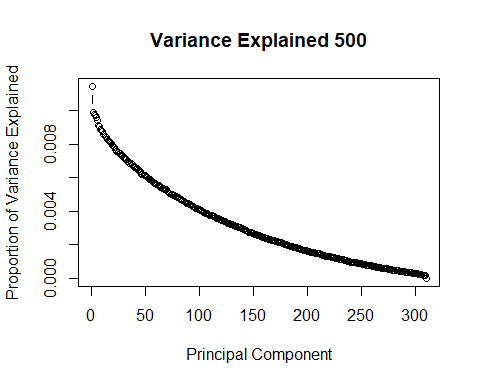
std\_dev <- prin\_comp$sdev  
pr\_var <- std\_dev^2  
pr\_var[1:10]

[1] 5.874800 5.064426 5.022195 4.982542 4.934381 4.843773 4.679750  
 [8] 4.662627 4.564496 4.523734

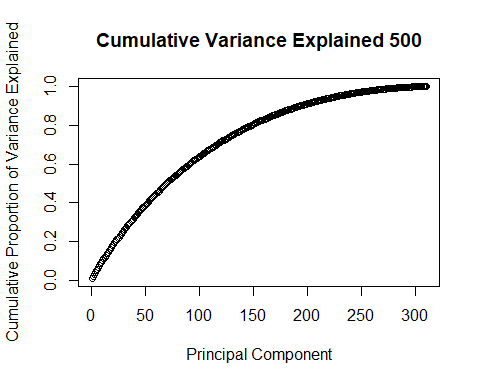
prop\_varex <- pr\_var/sum(pr\_var)  
prop\_varex[1:20]

[1] 0.011429572 0.009852969 0.009770806 0.009693662 0.009599963  
 [6] 0.009423684 0.009104571 0.009071258 0.008880343 0.008801038  
[11] 0.008727385 0.008548648 0.008463923 0.008390731 0.008292869  
[16] 0.008238491 0.008174125 0.008094779 0.008002274 0.007914800

plot(prop\_varex, xlab = "Principal Component",  
 ylab = "Proportion of Variance Explained",  
 type = "b", main='Variance Explained 500')



plot(cumsum(prop\_varex), xlab = "Principal Component",  
 ylab = "Cumulative Proportion of Variance Explained",  
 type = "b", main= "Cumulative Variance Explained 500")



# Add 1000 Random Variables

set.seed(12)  
df <- read.csv('Dataset\_spine.csv', col.names = c('pelvic\_incidence', 'pelvic\_tilt', 'lumbar\_lordosis\_angle', 'sacral\_slope', 'pelvic\_radius', 'degree\_spondylolisthesis', 'pelvic\_slope', 'Direct\_tilt', 'thoracic\_slope', 'cervical\_tilt', 'sacrum\_angle', 'scoliosis\_slope', 'classification', ' '))  
  
df$class = ifelse(df$classification == "Abnormal", 0, 1) #making classification numeric  
df$X. <- NULL #dropping the column with variable descriptions in it  
data = scale(df[, 1:12]) #scaling all except classification variable  
data = data.frame(data)  
data$classification = df$classification #add classification back into scaled dataframe  
data$class = df$class  
rand\_df = data.frame(matrix(rnorm(1000\*nrow(data)), nrow = nrow(data), ncol = 1000))  
data = cbind(data, rand\_df)  
  
  
data$rand\_int = runif(n=nrow(data), min = 0, max = 1) #random values from uniform distribution  
data2 = data[order(data$rand\_int),] #reorder dataframe by random value  
  
data2$group = seq(from = 1, to=5, by=1)  
data2$pred\_cv = 0  
  
detach(data2)  
data2$pred\_prob

NULL

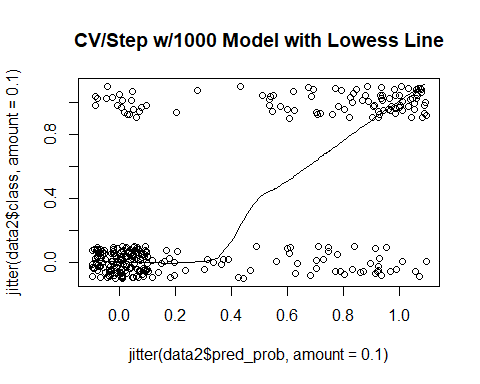
attach(data2)  
  
  
  
for (grp in 1:5){  
 train = data2[data2$group != grp, ]  
 test = data2[data2$group == grp,]  
   
 min.model = suppressWarnings(glm(class ~ 1, data=train, family=binomial(link="logit")))  
 biggest <- suppressWarnings(formula(glm(class ~ . - classification - pred\_cv - rand\_int - group, data = train, family=binomial(link="logit"))))  
  
 step.logit = step(min.model, direction='forward', scope=biggest, trace=0)  
 pred = predict(step.logit, train, type = 'response')  
 cutoffs = c()  
 accuracies = c()  
 for (val in as.vector(pred)){  
 cf = ModelMetrics::confusionMatrix(train$class, as.vector(pred), cutoff = val)  
 acc = (cf[1,1] + cf[2,2])/nrow(train)  
 accuracies = c(accuracies, acc)  
 cutoffs = c(cutoffs, val)  
 }  
  
 max1 = max(accuracies)  
 ind = match(max1, accuracies)  
 op = cutoffs[ind]  
   
 cv\_lr\_pred = predict(step.logit, test, type = 'response')  
 data2$pred\_prob[data2$group == grp] = cv\_lr\_pred  
 data2$pred\_cv[data2$group == grp] = ifelse(cv\_lr\_pred < op, 0, 1)  
}  
  
  
cv\_lr\_cfr\_w1000 = ModelMetrics::confusionMatrix( predicted = data2$pred\_cv, actual = data2$class)  
cv\_lr\_cfr\_w1000

[,1] [,2]  
[1,] 171 25  
[2,] 39 75

#accuracy score for Cross Validated RandomForest with 1000 Random Variables  
acc\_cv\_lr\_w1000 = (cv\_lr\_cfr\_w1000[1,1] + cv\_lr\_cfr\_w1000[2,2])/nrow(data2)  
acc\_cv\_lr\_w1000

[1] 0.7935484

plot(x = jitter(data2$pred\_prob, amount = .1), y= jitter(data2$class, amount = .1), main = "CV/Step w/1000 Model with Lowess Line")  
lines(lowess(x = jitter(data2$pred\_prob, amount = .1), y= jitter(data2$class, amount = .1)))



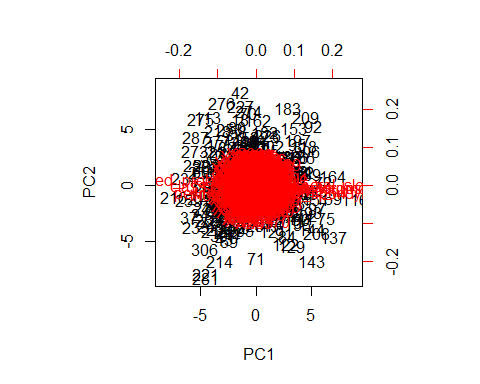
#PCA for 1000 random variables  
myvars <- names(data2) %in% c("classification", "rand\_int", "pred\_cv", "group") #remove classificaiton  
newdata = data2[!myvars]  
  
prin\_comp = prcomp(newdata, scale.= T)  
names(prin\_comp)

[1] "sdev" "rotation" "center" "scale" "x"

prin\_comp$rotation[1:5,1:4]

PC1 PC2 PC3 PC4  
pelvic\_incidence 0.26039490 -0.008145069 0.018005619 0.04386852  
pelvic\_tilt 0.15651824 -0.015057780 -0.042867289 0.01757024  
lumbar\_lordosis\_angle 0.22794649 -0.006828880 0.005227278 0.01130359  
sacral\_slope 0.21767067 0.000768123 0.055082960 0.04323084  
pelvic\_radius -0.09339356 -0.020479532 0.028754927 -0.01895133

biplot(prin\_comp, scale = 0)



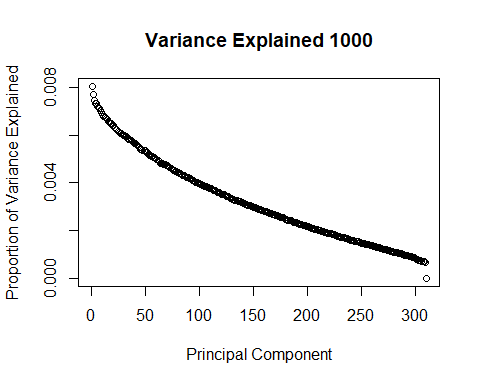
std\_dev <- prin\_comp$sdev  
pr\_var <- std\_dev^2  
pr\_var[1:10]

[1] 8.159674 7.822566 7.566269 7.419606 7.416358 7.311467 7.268683  
 [8] 7.240749 7.114971 7.056890

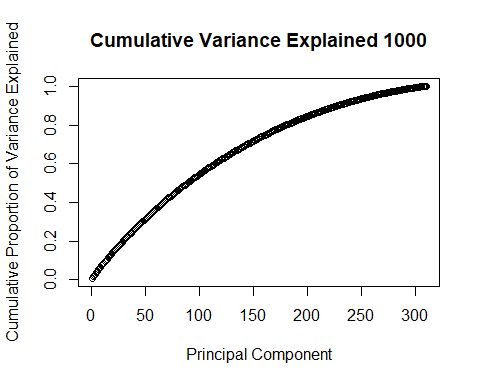
prop\_varex <- pr\_var/sum(pr\_var)  
prop\_varex[1:20]

[1] 0.008047015 0.007714562 0.007461803 0.007317166 0.007313962  
 [6] 0.007210520 0.007168326 0.007140778 0.007016737 0.006959457  
[11] 0.006859799 0.006795145 0.006750020 0.006689229 0.006664719  
[16] 0.006590777 0.006537566 0.006509526 0.006471214 0.006419935

plot(prop\_varex, xlab = "Principal Component",  
 ylab = "Proportion of Variance Explained",  
 type = "b", main='Variance Explained 1000')



plot(cumsum(prop\_varex), xlab = "Principal Component",  
 ylab = "Cumulative Proportion of Variance Explained",  
 type = "b", main= "Cumulative Variance Explained 1000")



Appendix III: Support Vector Machine

setwd("C:/Users/Michael Streyle/Desktop/Senior Project") #change this when i switch computers  
#setwd("C:/Users/Michael/Desktop/Senior Project")  
set.seed(12)  
  
data1 <- read.csv('Dataset\_spine.csv', col.names = c('pelvic\_incidence', 'pelvic\_tilt', 'lumbar\_lordosis\_angle', 'sacral\_slope', 'pelvic\_radius', 'degree\_spondylolisthesis', 'pelvic\_slope', 'Direct\_tilt', 'thoracic\_slope', 'cervical\_tilt', 'sacrum\_angle', 'scoliosis\_slope', 'classification', ' '))  
  
  
data1$X. <- NULL #dropping the column with variable descriptions in it  
data = scale(data1[, 1:12]) #scaling all except classification variable  
data = data.frame(data)  
data$classification = data1$classification #add classification back into scaled dataframe  
data$class = ifelse(data$classification == "Abnormal", 0, 1) #making classification numeric  
  
#these are the packages I use:  
  
# library(ModelMetrics) old use for confusion matrices. wasnt working for factor columns  
library(e1071) #SVM  
library(randomForest)

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

library(caret) #for confusion matrices

## Loading required package: lattice

## Loading required package: ggplot2

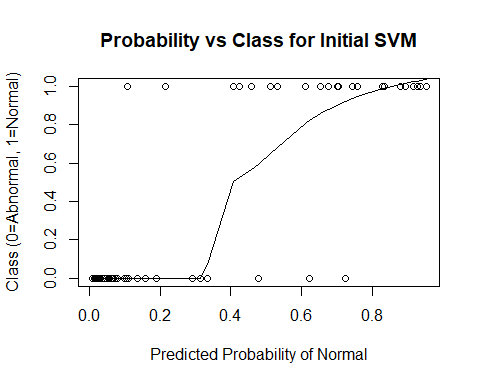
##   
## Attaching package: 'ggplot2'

## The following object is masked from 'package:randomForest':  
##   
## margin

attach(data)

# Initial Model

#make train and test set  
smp\_size <- floor(0.8 \* nrow(data))  
set.seed(12)  
train\_ind <- sample(seq\_len(nrow(data)), size = smp\_size)  
svm\_train <- data[train\_ind, ]  
svm\_test <- data[-train\_ind, ]  
  
#Fit a model. The function syntax is very similar to lm function  
   
model\_svm <- svm(classification ~ pelvic\_tilt + pelvic\_incidence + lumbar\_lordosis\_angle + sacral\_slope + pelvic\_radius + degree\_spondylolisthesis, data = svm\_train, probability = T)  
  
  
#Use the predictions on the data  
pred <- predict(model\_svm, svm\_test, probability = T)  
svm\_test$svm\_pred = pred  
  
plot(y=svm\_test$class, x=attr(svm\_test$svm\_pred, "probabilities")[,2], main = "Probability vs Class for Initial SVM", xlab = "Predicted Probability of Normal", ylab="Class (0=Abnormal, 1=Normal)")  
lines(lowess(y=svm\_test$class, x=attr(svm\_test$svm\_pred, "probabilities")[,2]))



svm\_cf = caret::confusionMatrix(data = svm\_test$svm\_pred, reference = svm\_test$classification)   
svm\_cf

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Abnormal Normal  
## Abnormal 38 5  
## Normal 2 17  
##   
## Accuracy : 0.8871   
## 95% CI : (0.7811, 0.9534)  
## No Information Rate : 0.6452   
## P-Value [Acc > NIR] : 1.515e-05   
##   
## Kappa : 0.7456   
## Mcnemar's Test P-Value : 0.4497   
##   
## Sensitivity : 0.9500   
## Specificity : 0.7727   
## Pos Pred Value : 0.8837   
## Neg Pred Value : 0.8947   
## Prevalence : 0.6452   
## Detection Rate : 0.6129   
## Detection Prevalence : 0.6935   
## Balanced Accuracy : 0.8614   
##   
## 'Positive' Class : Abnormal   
##

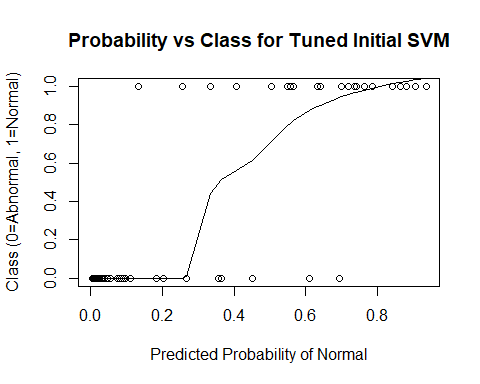
#tune initial model  
  
tuned\_parameters <- tune.svm(classification ~ pelvic\_tilt + pelvic\_incidence + lumbar\_lordosis\_angle + sacral\_slope + pelvic\_radius + degree\_spondylolisthesis, data = svm\_train, gamma = 10^(-5:-1), cost = 10^(-3:1))  
  
summary(tuned\_parameters )

##   
## Parameter tuning of 'svm':  
##   
## - sampling method: 10-fold cross validation   
##   
## - best parameters:  
## gamma cost  
## 0.1 1  
##   
## - best performance: 0.1323333   
##   
## - Detailed performance results:  
## gamma cost error dispersion  
## 1 1e-05 1e-03 0.3148333 0.06779804  
## 2 1e-04 1e-03 0.3148333 0.06779804  
## 3 1e-03 1e-03 0.3148333 0.06779804  
## 4 1e-02 1e-03 0.3148333 0.06779804  
## 5 1e-01 1e-03 0.3148333 0.06779804  
## 6 1e-05 1e-02 0.3148333 0.06779804  
## 7 1e-04 1e-02 0.3148333 0.06779804  
## 8 1e-03 1e-02 0.3148333 0.06779804  
## 9 1e-02 1e-02 0.3148333 0.06779804  
## 10 1e-01 1e-02 0.3148333 0.06779804  
## 11 1e-05 1e-01 0.3148333 0.06779804  
## 12 1e-04 1e-01 0.3148333 0.06779804  
## 13 1e-03 1e-01 0.3148333 0.06779804  
## 14 1e-02 1e-01 0.3148333 0.06779804  
## 15 1e-01 1e-01 0.2708333 0.07764771  
## 16 1e-05 1e+00 0.3148333 0.06779804  
## 17 1e-04 1e+00 0.3148333 0.06779804  
## 18 1e-03 1e+00 0.3148333 0.06779804  
## 19 1e-02 1e+00 0.2266667 0.13152712  
## 20 1e-01 1e+00 0.1323333 0.05612596  
## 21 1e-05 1e+01 0.3148333 0.06779804  
## 22 1e-04 1e+01 0.3148333 0.06779804  
## 23 1e-03 1e+01 0.2265000 0.12364438  
## 24 1e-02 1e+01 0.1405000 0.06538608  
## 25 1e-01 1e+01 0.1363333 0.06254282

tune\_model\_svm <- svm(classification ~ pelvic\_tilt + pelvic\_incidence + lumbar\_lordosis\_angle + sacral\_slope + pelvic\_radius + degree\_spondylolisthesis, data = svm\_train, gamma = 0.1, cost= 1, probability=T)  
  
pred2 <- predict(tune\_model\_svm, svm\_test, probability = T)  
svm\_test$svm\_pred2 = pred2  
  
  
tune\_svm\_cf = caret::confusionMatrix(data = svm\_test$svm\_pred2, reference = svm\_test$classification)   
tune\_svm\_cf

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Abnormal Normal  
## Abnormal 38 4  
## Normal 2 18  
##   
## Accuracy : 0.9032   
## 95% CI : (0.8012, 0.9637)  
## No Information Rate : 0.6452   
## P-Value [Acc > NIR] : 3.301e-06   
##   
## Kappa : 0.7842   
## Mcnemar's Test P-Value : 0.6831   
##   
## Sensitivity : 0.9500   
## Specificity : 0.8182   
## Pos Pred Value : 0.9048   
## Neg Pred Value : 0.9000   
## Prevalence : 0.6452   
## Detection Rate : 0.6129   
## Detection Prevalence : 0.6774   
## Balanced Accuracy : 0.8841   
##   
## 'Positive' Class : Abnormal   
##

plot(y=svm\_test$class, x=attr(svm\_test$svm\_pred2, "probabilities")[,2], main = "Probability vs Class for Tuned Initial SVM", xlab = "Predicted Probability of Normal", ylab="Class (0=Abnormal, 1=Normal)")  
lines(lowess(y=svm\_test$class, x=attr(svm\_test$svm\_pred2, "probabilities")[,2]))



# Initial Model with Cross-Validation

set.seed(12)  
data$rand\_int = runif(n=nrow(data), min = 0, max = 1) #random values from uniform distribution  
data2 = data[order(data$rand\_int),] #reorder dataframe by random value  
  
data2$group = seq(from = 1, to=5, by=1)  
  
attach(data2)

## The following objects are masked from data:  
##   
## cervical\_tilt, class, classification,  
## degree\_spondylolisthesis, Direct\_tilt, lumbar\_lordosis\_angle,  
## pelvic\_incidence, pelvic\_radius, pelvic\_slope, pelvic\_tilt,  
## sacral\_slope, sacrum\_angle, scoliosis\_slope, thoracic\_slope

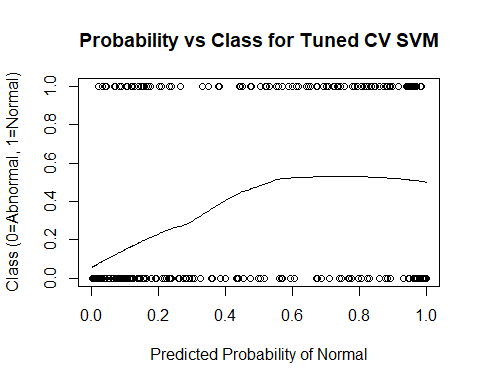
data2$pred\_svm = 0  
data2$pred\_class = as.factor(c("Abnormal", "Normal"))  
#find best parameters  
tuned = tune.svm(classification ~ pelvic\_tilt + pelvic\_incidence + lumbar\_lordosis\_angle +   
 sacral\_slope + pelvic\_radius + degree\_spondylolisthesis, data = data2, gamma = 10^(-5:-1), cost = 10^(-3:1), tunecontrol=tune.control(cross=5))  
tuned$best.parameters

## gamma cost  
## 25 0.1 10

#train with cross-validation  
for (grp in 1:5){  
 train = data2[data2$group != grp, ]  
 test = data2[data2$group == grp,]  
   
 cv\_svm <- svm(classification ~ pelvic\_tilt + pelvic\_incidence + lumbar\_lordosis\_angle +   
 sacral\_slope + pelvic\_radius + degree\_spondylolisthesis, data = train, gamma=0.1, cost=10, probability = T)  
  
 cv\_svm\_pred = attr(predict(cv\_svm, test, probability = T), "probabilities")[,1]  
 data2$pred\_svm[data2$group == grp] = cv\_svm\_pred  
   
 cv\_svm\_class = predict(cv\_svm, test)  
 data2$pred\_class[data2$group == grp] = cv\_svm\_class  
}  
  
  
cv\_svm\_cfr = caret::confusionMatrix( data = data2$pred\_class, reference = data2$classification)  
cv\_svm\_cfr

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Abnormal Normal  
## Abnormal 186 23  
## Normal 24 77  
##   
## Accuracy : 0.8484   
## 95% CI : (0.8035, 0.8864)  
## No Information Rate : 0.6774   
## P-Value [Acc > NIR] : 5.083e-12   
##   
## Kappa : 0.654   
## Mcnemar's Test P-Value : 1   
##   
## Sensitivity : 0.8857   
## Specificity : 0.7700   
## Pos Pred Value : 0.8900   
## Neg Pred Value : 0.7624   
## Prevalence : 0.6774   
## Detection Rate : 0.6000   
## Detection Prevalence : 0.6742   
## Balanced Accuracy : 0.8279   
##   
## 'Positive' Class : Abnormal   
##

plot(y=data2$class, x=data2$pred\_svm, main = "Probability vs Class for Tuned CV SVM", xlab = "Predicted Probability of Normal", ylab="Class (0=Abnormal, 1=Normal)")  
lines(lowess(y=data2$class, x=data2$pred\_svm))



# Add Provided Random Noise

set.seed(12)  
df <- read.csv('Dataset\_spine.csv', col.names = c('pelvic\_incidence', 'pelvic\_tilt', 'lumbar\_lordosis\_angle', 'sacral\_slope', 'pelvic\_radius', 'degree\_spondylolisthesis', 'pelvic\_slope', 'Direct\_tilt', 'thoracic\_slope', 'cervical\_tilt', 'sacrum\_angle', 'scoliosis\_slope', 'classification', ' '))  
  
df$class = ifelse(df$classification == "Abnormal", 0, 1) #making classification numeric  
df$X. <- NULL #dropping the column with variable descriptions in it  
  
  
  
data = scale(df[, 1:12]) #scaling all except classification variable  
data = data.frame(data)  
data$classification = df$classification #add classification back into scaled dataframe  
data$class = df$class  
set.seed(12)  
data$rand\_int = runif(n=nrow(data), min = 0, max = 1) #random values from uniform distribution  
data2 = data[order(data$rand\_int),] #reorder dataframe by random value  
  
data2$group = seq(from = 1, to=5, by=1)  
data2$pred\_class = as.factor(c('Abnormal', 'Normal'))  
data2$pred\_svm =0  
attach(data2)

## The following objects are masked from data2 (pos = 3):  
##   
## cervical\_tilt, class, classification,  
## degree\_spondylolisthesis, Direct\_tilt, group,  
## lumbar\_lordosis\_angle, pelvic\_incidence, pelvic\_radius,  
## pelvic\_slope, pelvic\_tilt, rand\_int, sacral\_slope,  
## sacrum\_angle, scoliosis\_slope, thoracic\_slope

## The following objects are masked from data:  
##   
## cervical\_tilt, class, classification,  
## degree\_spondylolisthesis, Direct\_tilt, lumbar\_lordosis\_angle,  
## pelvic\_incidence, pelvic\_radius, pelvic\_slope, pelvic\_tilt,  
## sacral\_slope, sacrum\_angle, scoliosis\_slope, thoracic\_slope

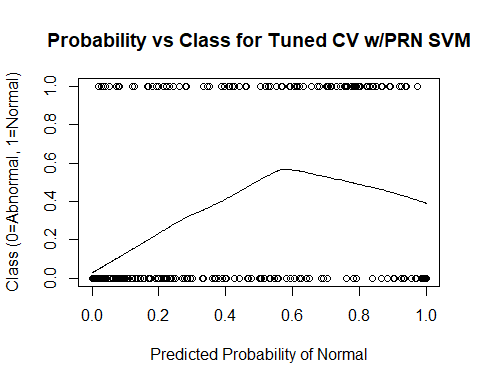
tuned = tune.svm(classification ~ . - class - rand\_int - group - pred\_class - pred\_svm, data = data2, gamma = 10^(-5:-1), cost = 10^(-3:1), tunecontrol=tune.control(cross=5))  
tuned$best.parameters

## gamma cost  
## 24 0.01 10

for (grp in 1:5){  
 train = data2[data2$group != grp, ]  
 test = data2[data2$group == grp,]  
   
 svm <- svm(classification ~ . - class - rand\_int - group - pred\_class - pred\_svm, gamma=0.01, cost=10,  
 data=train, probability = T)  
 svm\_pred = attr(predict(svm, test, probability = T), "probabilities")[,1]  
 data2$pred\_svm[data2$group == grp] = svm\_pred  
   
 svm\_class = predict(svm, test)  
 data2$pred\_class[data2$group == grp] = svm\_class  
}  
  
cv\_svm\_cfr\_wprn = caret::confusionMatrix(data2$pred\_class,data2$classification)  
cv\_svm\_cfr\_wprn #with provided random noise

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Abnormal Normal  
## Abnormal 184 27  
## Normal 26 73  
##   
## Accuracy : 0.829   
## 95% CI : (0.7824, 0.8692)  
## No Information Rate : 0.6774   
## P-Value [Acc > NIR] : 1.229e-09   
##   
## Kappa : 0.6078   
## Mcnemar's Test P-Value : 1   
##   
## Sensitivity : 0.8762   
## Specificity : 0.7300   
## Pos Pred Value : 0.8720   
## Neg Pred Value : 0.7374   
## Prevalence : 0.6774   
## Detection Rate : 0.5935   
## Detection Prevalence : 0.6806   
## Balanced Accuracy : 0.8031   
##   
## 'Positive' Class : Abnormal   
##

plot(y=data2$class, x=data2$pred\_svm, main = "Probability vs Class for Tuned CV w/PRN SVM", xlab = "Predicted Probability of Normal", ylab="Class (0=Abnormal, 1=Normal)")  
lines(lowess(y=data2$class, x=data2$pred\_svm))



# Add 10 Random Variables

df <- read.csv('Dataset\_spine.csv', col.names = c('pelvic\_incidence', 'pelvic\_tilt', 'lumbar\_lordosis\_angle', 'sacral\_slope', 'pelvic\_radius', 'degree\_spondylolisthesis', 'pelvic\_slope', 'Direct\_tilt', 'thoracic\_slope', 'cervical\_tilt', 'sacrum\_angle', 'scoliosis\_slope', 'classification', ' '))  
  
df$class = ifelse(df$classification == "Abnormal", 0, 1) #making classification numeric  
df$X. <- NULL #dropping the column with variable descriptions in it  
data = scale(df[, 1:12]) #scaling all except classification variable  
data = data.frame(data)  
data$classification = df$classification #add classification back into scaled dataframe  
data$class = df$class  
set.seed(12)  
rand\_df = data.frame(matrix(rnorm(10\*nrow(data) ), nrow = nrow(data), ncol = 10))  
data = cbind(data, rand\_df)  
  
  
data$rand\_int = runif(n=nrow(data), min = 0, max = 1) #random values from uniform distribution  
data2 = data[order(data$rand\_int),] #reorder dataframe by random value  
  
data2$group = seq(from = 1, to=5, by=1)  
data2$pred\_class = as.factor(c('Abnormal', 'Normal'))  
data2$pred\_svm =0  
attach(data2)

## The following objects are masked from data2 (pos = 3):  
##   
## cervical\_tilt, class, classification,  
## degree\_spondylolisthesis, Direct\_tilt, group,  
## lumbar\_lordosis\_angle, pelvic\_incidence, pelvic\_radius,  
## pelvic\_slope, pelvic\_tilt, pred\_class, pred\_svm, rand\_int,  
## sacral\_slope, sacrum\_angle, scoliosis\_slope, thoracic\_slope

## The following objects are masked from data2 (pos = 4):  
##   
## cervical\_tilt, class, classification,  
## degree\_spondylolisthesis, Direct\_tilt, group,  
## lumbar\_lordosis\_angle, pelvic\_incidence, pelvic\_radius,  
## pelvic\_slope, pelvic\_tilt, rand\_int, sacral\_slope,  
## sacrum\_angle, scoliosis\_slope, thoracic\_slope

## The following objects are masked from data:  
##   
## cervical\_tilt, class, classification,  
## degree\_spondylolisthesis, Direct\_tilt, lumbar\_lordosis\_angle,  
## pelvic\_incidence, pelvic\_radius, pelvic\_slope, pelvic\_tilt,  
## sacral\_slope, sacrum\_angle, scoliosis\_slope, thoracic\_slope

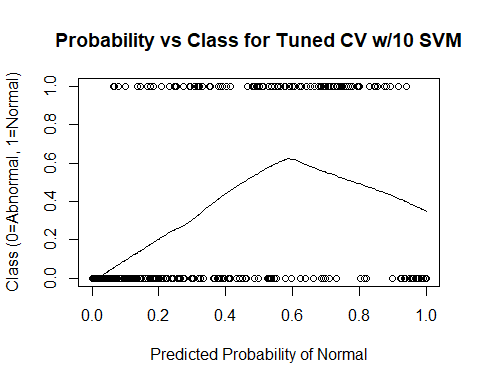
tuned = tune.svm(classification ~ . - class - rand\_int - group - pred\_class - pred\_svm, data = data2, gamma = 10^(-5:-1), cost = 10^(-3:1), tunecontrol=tune.control(cross=5))  
tuned$best.parameters

## gamma cost  
## 24 0.01 10

for (grp in 1:5){  
 train = data2[data2$group != grp, ]  
 test = data2[data2$group == grp,]  
   
 svm <- svm(classification ~ . - class - rand\_int - group - pred\_class - pred\_svm, gamma=0.01, cost=10,  
 data=train, probability = T)  
 svm\_pred = attr(predict(svm, test, probability = T), "probabilities")[,2]  
 data2$pred\_svm[data2$group == grp] = svm\_pred  
   
 svm\_class = predict(svm, test)  
 data2$pred\_class[data2$group == grp] = svm\_class  
}  
  
cv\_svm\_cfr\_w10 = caret::confusionMatrix(data2$pred\_class,data2$classification)  
cv\_svm\_cfr\_w10 #with provided random noise

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Abnormal Normal  
## Abnormal 181 29  
## Normal 29 71  
##   
## Accuracy : 0.8129   
## 95% CI : (0.765, 0.8548)  
## No Information Rate : 0.6774   
## P-Value [Acc > NIR] : 6.441e-08   
##   
## Kappa : 0.5719   
## Mcnemar's Test P-Value : 1   
##   
## Sensitivity : 0.8619   
## Specificity : 0.7100   
## Pos Pred Value : 0.8619   
## Neg Pred Value : 0.7100   
## Prevalence : 0.6774   
## Detection Rate : 0.5839   
## Detection Prevalence : 0.6774   
## Balanced Accuracy : 0.7860   
##   
## 'Positive' Class : Abnormal   
##

plot(y=data2$class, x=data2$pred\_svm, main = "Probability vs Class for Tuned CV w/10 SVM", xlab = "Predicted Probability of Normal", ylab="Class (0=Abnormal, 1=Normal)")  
lines(lowess(y=data2$class, x=data2$pred\_svm))



# Add 100 Random Variables

set.seed(12)  
df <- read.csv('Dataset\_spine.csv', col.names = c('pelvic\_incidence', 'pelvic\_tilt', 'lumbar\_lordosis\_angle', 'sacral\_slope', 'pelvic\_radius', 'degree\_spondylolisthesis', 'pelvic\_slope', 'Direct\_tilt', 'thoracic\_slope', 'cervical\_tilt', 'sacrum\_angle', 'scoliosis\_slope', 'classification', ' '))  
  
df$class = ifelse(df$classification == "Abnormal", 0, 1) #making classification numeric  
df$X. <- NULL #dropping the column with variable descriptions in it  
data = scale(df[, 1:12]) #scaling all except classification variable  
data = data.frame(data)  
data$classification = df$classification #add classification back into scaled dataframe  
data$class = df$class  
set.seed(12)  
rand\_df = data.frame(matrix(rnorm(100\*nrow(data) ), nrow = nrow(data), ncol = 100))  
data = cbind(data, rand\_df)  
  
  
data$rand\_int = runif(n=nrow(data), min = 0, max = 1) #random values from uniform distribution  
data2 = data[order(data$rand\_int),] #reorder dataframe by random value  
  
data2$group = seq(from = 1, to=5, by=1)  
data2$pred\_class = as.factor(c('Abnormal', 'Normal'))  
data2$pred\_svm =0  
attach(data2)

## The following objects are masked from data2 (pos = 3):  
##   
## cervical\_tilt, class, classification,  
## degree\_spondylolisthesis, Direct\_tilt, group,  
## lumbar\_lordosis\_angle, pelvic\_incidence, pelvic\_radius,  
## pelvic\_slope, pelvic\_tilt, pred\_class, pred\_svm, rand\_int,  
## sacral\_slope, sacrum\_angle, scoliosis\_slope, thoracic\_slope,  
## X1, X10, X2, X3, X4, X5, X6, X7, X8, X9

## The following objects are masked from data2 (pos = 4):  
##   
## cervical\_tilt, class, classification,  
## degree\_spondylolisthesis, Direct\_tilt, group,  
## lumbar\_lordosis\_angle, pelvic\_incidence, pelvic\_radius,  
## pelvic\_slope, pelvic\_tilt, pred\_class, pred\_svm, rand\_int,  
## sacral\_slope, sacrum\_angle, scoliosis\_slope, thoracic\_slope

## The following objects are masked from data2 (pos = 5):  
##   
## cervical\_tilt, class, classification,  
## degree\_spondylolisthesis, Direct\_tilt, group,  
## lumbar\_lordosis\_angle, pelvic\_incidence, pelvic\_radius,  
## pelvic\_slope, pelvic\_tilt, rand\_int, sacral\_slope,  
## sacrum\_angle, scoliosis\_slope, thoracic\_slope

## The following objects are masked from data:  
##   
## cervical\_tilt, class, classification,  
## degree\_spondylolisthesis, Direct\_tilt, lumbar\_lordosis\_angle,  
## pelvic\_incidence, pelvic\_radius, pelvic\_slope, pelvic\_tilt,  
## sacral\_slope, sacrum\_angle, scoliosis\_slope, thoracic\_slope

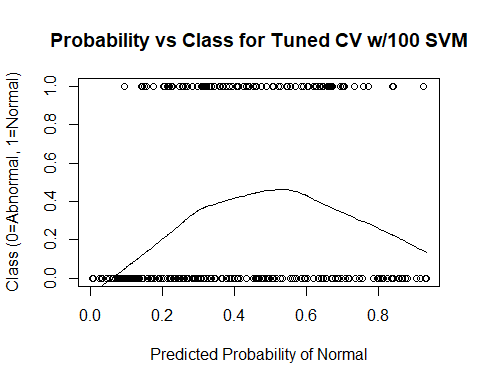
tuned = tune.svm(classification ~ . - class - rand\_int - group - pred\_class - pred\_svm, data = data2, gamma = 10^(-5:-1), cost = 10^(-3:1), tunecontrol=tune.control(cross=5))  
tuned$best.parameters

## gamma cost  
## 23 0.001 10

for (grp in 1:5){  
 train = data2[data2$group != grp, ]  
 test = data2[data2$group == grp,]  
   
 svm <- svm(classification ~ . - class - rand\_int - group - pred\_class - pred\_svm, gamma=0.001, cost=10,  
 data=train, probability = T)  
 svm\_pred = attr(predict(svm, test, probability = T), "probabilities")[,2]  
 data2$pred\_svm[data2$group == grp] = svm\_pred  
   
 svm\_class = predict(svm, test)  
 data2$pred\_class[data2$group == grp] = svm\_class  
}  
  
cv\_svm\_cfr\_w100 = caret::confusionMatrix(data2$pred\_class,data2$classification)  
cv\_svm\_cfr\_w100

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Abnormal Normal  
## Abnormal 169 55  
## Normal 41 45  
##   
## Accuracy : 0.6903   
## 95% CI : (0.6356, 0.7414)  
## No Information Rate : 0.6774   
## P-Value [Acc > NIR] : 0.3375   
##   
## Kappa : 0.2645   
## Mcnemar's Test P-Value : 0.1846   
##   
## Sensitivity : 0.8048   
## Specificity : 0.4500   
## Pos Pred Value : 0.7545   
## Neg Pred Value : 0.5233   
## Prevalence : 0.6774   
## Detection Rate : 0.5452   
## Detection Prevalence : 0.7226   
## Balanced Accuracy : 0.6274   
##   
## 'Positive' Class : Abnormal   
##

plot(y=data2$class, x=data2$pred\_svm, main = "Probability vs Class for Tuned CV w/100 SVM", xlab = "Predicted Probability of Normal", ylab="Class (0=Abnormal, 1=Normal)")  
lines(lowess(y=data2$class, x=data2$pred\_svm))



# Add 500 Random Variables

set.seed(12)  
df <- read.csv('Dataset\_spine.csv', col.names = c('pelvic\_incidence', 'pelvic\_tilt', 'lumbar\_lordosis\_angle', 'sacral\_slope', 'pelvic\_radius', 'degree\_spondylolisthesis', 'pelvic\_slope', 'Direct\_tilt', 'thoracic\_slope', 'cervical\_tilt', 'sacrum\_angle', 'scoliosis\_slope', 'classification', ' '))  
  
df$class = ifelse(df$classification == "Abnormal", 0, 1) #making classification numeric  
df$X. <- NULL #dropping the column with variable descriptions in it  
data = scale(df[, 1:12]) #scaling all except classification variable  
data = data.frame(data)  
data$classification = df$classification #add classification back into scaled dataframe  
data$class = df$class  
set.seed(12)  
rand\_df = data.frame(matrix(rnorm(500\*nrow(data) ), nrow = nrow(data), ncol = 500))  
data = cbind(data, rand\_df)  
  
  
data$rand\_int = runif(n=nrow(data), min = 0, max = 1) #random values from uniform distribution  
data2 = data[order(data$rand\_int),] #reorder dataframe by random value  
  
data2$group = seq(from = 1, to=5, by=1)  
data2$pred\_class = as.factor(c('Abnormal', 'Normal'))  
data2$pred\_svm =0  
attach(data2)

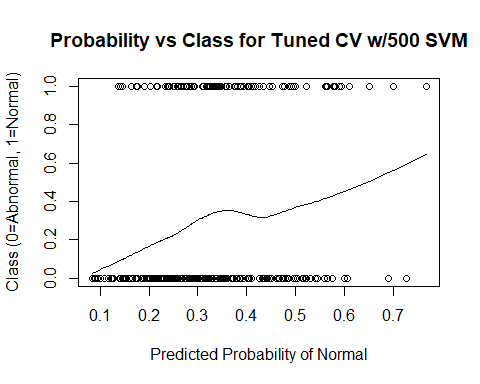
tuned = tune.svm(classification ~ . - class - rand\_int - group - pred\_class - pred\_svm, data = data2, gamma = 10^(-5:-1), cost = 10^(-3:1), tunecontrol=tune.control(cross=5))  
tuned$best.parameters

## gamma cost  
## 1 1e-05 0.001

for (grp in 1:5){  
 train = data2[data2$group != grp, ]  
 test = data2[data2$group == grp,]  
   
 svm <- svm(classification ~ . - class - rand\_int - group - pred\_class - pred\_svm, gamma=1e-05, cost=0.001,  
 data=train, probability = T)  
 svm\_pred = attr(predict(svm, test, probability = T), "probabilities")[,1]  
 data2$pred\_svm[data2$group == grp] = svm\_pred  
   
 svm\_class = predict(svm, test)  
 data2$pred\_class[data2$group == grp] = svm\_class  
}  
  
cv\_svm\_cfr\_w500 = caret::confusionMatrix(data2$pred\_class,data2$classification)  
cv\_svm\_cfr\_w500

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Abnormal Normal  
## Abnormal 210 100  
## Normal 0 0  
##   
## Accuracy : 0.6774   
## 95% CI : (0.6223, 0.7292)  
## No Information Rate : 0.6774   
## P-Value [Acc > NIR] : 0.5271   
##   
## Kappa : 0   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 1.0000   
## Specificity : 0.0000   
## Pos Pred Value : 0.6774   
## Neg Pred Value : NaN   
## Prevalence : 0.6774   
## Detection Rate : 0.6774   
## Detection Prevalence : 1.0000   
## Balanced Accuracy : 0.5000   
##   
## 'Positive' Class : Abnormal   
##

plot(y=data2$class, x=data2$pred\_svm, main = "Probability vs Class for Tuned CV w/500 SVM", xlab = "Predicted Probability of Normal", ylab="Class (0=Abnormal, 1=Normal)")  
lines(lowess(y=data2$class, x=data2$pred\_svm))



# Add 1000 Random Variables

set.seed(12)  
df <- read.csv('Dataset\_spine.csv', col.names = c('pelvic\_incidence', 'pelvic\_tilt', 'lumbar\_lordosis\_angle', 'sacral\_slope', 'pelvic\_radius', 'degree\_spondylolisthesis', 'pelvic\_slope', 'Direct\_tilt', 'thoracic\_slope', 'cervical\_tilt', 'sacrum\_angle', 'scoliosis\_slope', 'classification', ' '))  
  
df$class = ifelse(df$classification == "Abnormal", 0, 1) #making classification numeric  
df$X. <- NULL #dropping the column with variable descriptions in it  
data = scale(df[, 1:12]) #scaling all except classification variable  
data = data.frame(data)  
data$classification = df$classification #add classification back into scaled dataframe  
data$class = df$class  
set.seed(12)  
rand\_df = data.frame(matrix(rnorm(1000\*nrow(data) ), nrow = nrow(data), ncol = 1000))  
data = cbind(data, rand\_df)  
  
  
data$rand\_int = runif(n=nrow(data), min = 0, max = 1) #random values from uniform distribution  
data2 = data[order(data$rand\_int),] #reorder dataframe by random value  
  
data2$group = seq(from = 1, to=5, by=1)  
data2$pred\_class = as.factor(c('Abnormal', 'Normal'))  
data2$pred\_svm =0  
attach(data2)

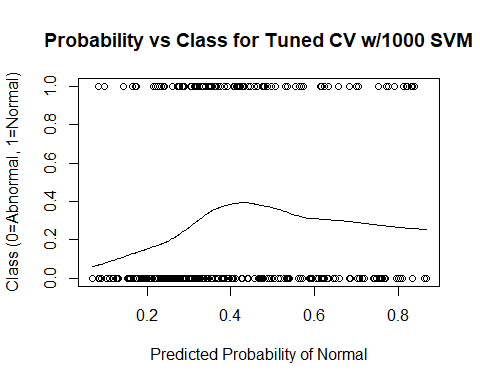
tuned = tune.svm(classification ~ . - class - rand\_int - group - pred\_class - pred\_svm, data = data2, gamma = 10^(-5:-1), cost = 10^(-3:1), tunecontrol=tune.control(cross=5))  
tuned$best.parameters

## gamma cost  
## 23 0.001 10

for (grp in 1:5){  
 train = data2[data2$group != grp, ]  
 test = data2[data2$group == grp,]  
   
 svm <- svm(classification ~ . - class - rand\_int - group - pred\_class - pred\_svm, gamma=0.001, cost=10,  
 data=train, probability = T)  
 svm\_pred = attr(predict(svm, test, probability = T), "probabilities")[,1]  
 data2$pred\_svm[data2$group == grp] = svm\_pred  
   
 svm\_class = predict(svm, test)  
 data2$pred\_class[data2$group == grp] = svm\_class  
}  
  
cv\_svm\_cfr\_w1000 = caret::confusionMatrix(data2$pred\_class,data2$classification)  
cv\_svm\_cfr\_w1000

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Abnormal Normal  
## Abnormal 209 95  
## Normal 1 5  
##   
## Accuracy : 0.6903   
## 95% CI : (0.6356, 0.7414)  
## No Information Rate : 0.6774   
## P-Value [Acc > NIR] : 0.3375   
##   
## Kappa : 0.06   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.9952   
## Specificity : 0.0500   
## Pos Pred Value : 0.6875   
## Neg Pred Value : 0.8333   
## Prevalence : 0.6774   
## Detection Rate : 0.6742   
## Detection Prevalence : 0.9806   
## Balanced Accuracy : 0.5226   
##   
## 'Positive' Class : Abnormal   
##

plot(y=data2$class, x=data2$pred\_svm, main = "Probability vs Class for Tuned CV w/1000 SVM", xlab = "Predicted Probability of Normal", ylab="Class (0=Abnormal, 1=Normal)")  
lines(lowess(y=data2$class, x=data2$pred\_svm))



Appendix IV: Random Forest

# Introduction

setwd("C:/Users/Michael Streyle/Desktop/Senior Project") #change this when i switch computers  
#setwd("C:/Users/Michael/Desktop/Senior Project")  
  
data1 <- read.csv('Dataset\_spine.csv', col.names = c('pelvic\_incidence', 'pelvic\_tilt', 'lumbar\_lordosis\_angle', 'sacral\_slope', 'pelvic\_radius', 'degree\_spondylolisthesis', 'pelvic\_slope', 'Direct\_tilt', 'thoracic\_slope', 'cervical\_tilt', 'sacrum\_angle', 'scoliosis\_slope', 'classification', ' '))  
  
  
data1$X. <- NULL #dropping the column with variable descriptions in it  
  
  
  
data = scale(data1[, 1:12]) #scaling all except classification variable  
data = data.frame(data)  
data$classification = data1$classification #add classification back into scaled dataframe  
data$class = ifelse(data$classification == "Abnormal", 0, 1) #making classification numeric  
  
#these are the packages I use:  
  
# library(ModelMetrics) old use for confusion matrices. wasnt working for factor columns  
library(e1071) #SVM  
library(randomForest)

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

library(caret) #for confusion matrices

## Loading required package: lattice

## Loading required package: ggplot2

##   
## Attaching package: 'ggplot2'

## The following object is masked from 'package:randomForest':  
##   
## margin

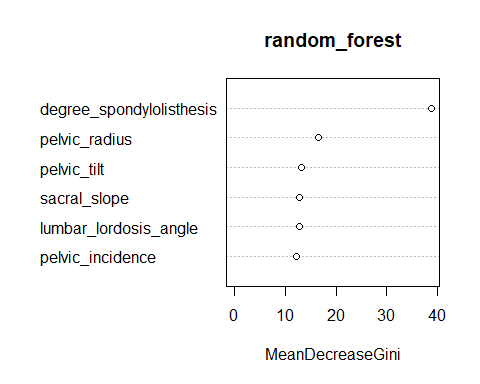
attach(data)

# Initial Model

#make train and test set  
smp\_size <- floor(0.8 \* nrow(data))  
set.seed(12)  
train\_ind <- sample(seq\_len(nrow(data)), size = smp\_size)  
rf\_train <- data[train\_ind, ]  
rf\_test <- data[-train\_ind, ]  
  
  
random\_forest = randomForest(classification ~ pelvic\_tilt + pelvic\_incidence + lumbar\_lordosis\_angle + sacral\_slope + pelvic\_radius + degree\_spondylolisthesis, data = rf\_train)  
  
#importance of variables  
importance(random\_forest)

## MeanDecreaseGini  
## pelvic\_tilt 13.15211  
## pelvic\_incidence 12.14187  
## lumbar\_lordosis\_angle 12.68480  
## sacral\_slope 12.80926  
## pelvic\_radius 16.58574  
## degree\_spondylolisthesis 38.84536

varImpPlot(random\_forest,type=2)

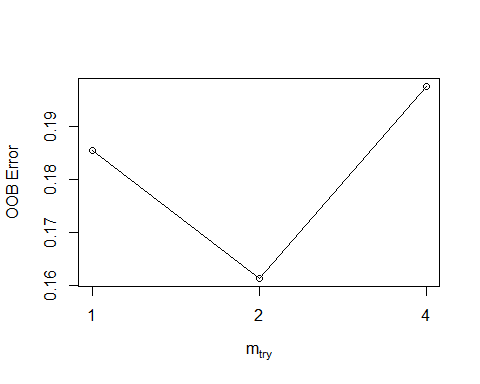


pred = predict(random\_forest, rf\_test)  
rf\_test$rf\_pred = pred  
  
rf\_cf = caret::confusionMatrix(rf\_test$classification, rf\_test$rf\_pred)  
rf\_cf

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Abnormal Normal  
## Abnormal 38 2  
## Normal 10 12  
##   
## Accuracy : 0.8065   
## 95% CI : (0.6863, 0.8958)  
## No Information Rate : 0.7742   
## P-Value [Acc > NIR] : 0.33265   
##   
## Kappa : 0.5396   
## Mcnemar's Test P-Value : 0.04331   
##   
## Sensitivity : 0.7917   
## Specificity : 0.8571   
## Pos Pred Value : 0.9500   
## Neg Pred Value : 0.5455   
## Prevalence : 0.7742   
## Detection Rate : 0.6129   
## Detection Prevalence : 0.6452   
## Balanced Accuracy : 0.8244   
##   
## 'Positive' Class : Abnormal   
##

#accuracy score is now in the output of the classification matrix.   
  
im = tuneRF(x = rf\_train[,1:6], y = rf\_train[,13], doBest = T)

## mtry = 2 OOB error = 16.13%   
## Searching left ...  
## mtry = 1 OOB error = 18.55%   
## -0.15 0.05   
## Searching right ...  
## mtry = 4 OOB error = 19.76%   
## -0.225 0.05



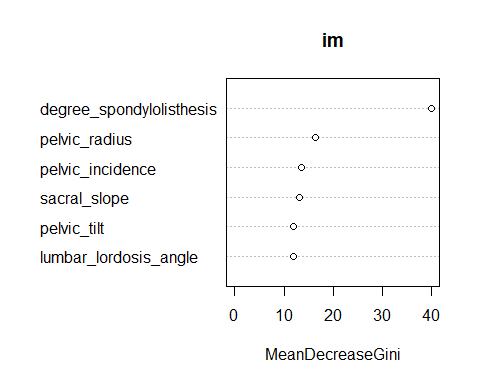
im

##   
## Call:  
## randomForest(x = x, y = y, mtry = res[which.min(res[, 2]), 1])   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 2  
##   
## OOB estimate of error rate: 16.94%  
## Confusion matrix:  
## Abnormal Normal class.error  
## Abnormal 152 18 0.1058824  
## Normal 24 54 0.3076923

importance(im)

## MeanDecreaseGini  
## pelvic\_incidence 13.60084  
## pelvic\_tilt 11.98641  
## lumbar\_lordosis\_angle 11.80908  
## sacral\_slope 13.10558  
## pelvic\_radius 16.39389  
## degree\_spondylolisthesis 39.96065

varImpPlot(im,type=2)



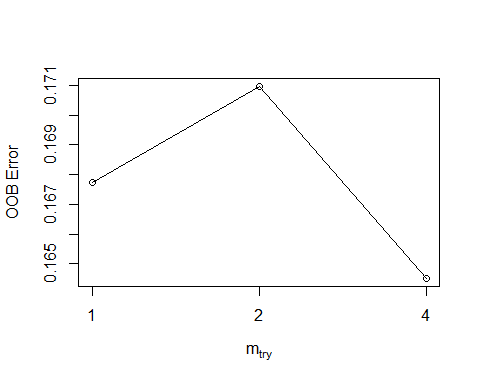
# Cross-Validation Model

data$rand\_int = runif(n=nrow(data), min = 0, max = 1) #random values from uniform distribution  
data2 = data[order(data$rand\_int),] #reorder dataframe by random value  
  
data2$group = seq(from = 1, to=5, by=1)  
data2$pred\_ranfor = 0  
data2$pred\_class = as.factor(x = c("Abnormal", "Normal"))  
attach(data2)

## The following objects are masked from data:  
##   
## cervical\_tilt, class, classification,  
## degree\_spondylolisthesis, Direct\_tilt, lumbar\_lordosis\_angle,  
## pelvic\_incidence, pelvic\_radius, pelvic\_slope, pelvic\_tilt,  
## sacral\_slope, sacrum\_angle, scoliosis\_slope, thoracic\_slope

im\_cv = tuneRF(x = data2[,1:6], y = data2[,13])

## mtry = 2 OOB error = 17.1%   
## Searching left ...  
## mtry = 1 OOB error = 16.77%   
## 0.01886792 0.05   
## Searching right ...  
## mtry = 4 OOB error = 16.45%   
## 0.03773585 0.05



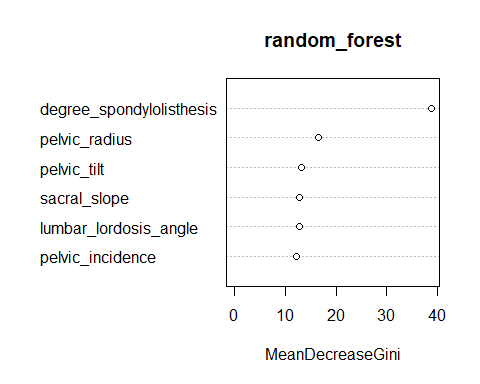
im\_cv

## mtry OOBError  
## 1.OOB 1 0.1677419  
## 2.OOB 2 0.1709677  
## 4.OOB 4 0.1645161

for (grp in 1:5){  
 train = data2[data2$group != grp, ]  
 test = data2[data2$group == grp,]  
   
 cv\_ranfor <- randomForest(classification ~ pelvic\_tilt + pelvic\_incidence + lumbar\_lordosis\_angle +   
 sacral\_slope + pelvic\_radius + degree\_spondylolisthesis, data = train, mtry=2)  
  
 pred\_ranfor = predict(object = cv\_ranfor, test, "prob")[,"Normal"]  
 data2$pred\_ranfor[data2$group == grp] = pred\_ranfor  
   
 pred\_class = predict(object = cv\_ranfor, test)  
 data2$pred\_class[data2$group == grp] = pred\_class  
}  
  
importance(random\_forest)

## MeanDecreaseGini  
## pelvic\_tilt 13.15211  
## pelvic\_incidence 12.14187  
## lumbar\_lordosis\_angle 12.68480  
## sacral\_slope 12.80926  
## pelvic\_radius 16.58574  
## degree\_spondylolisthesis 38.84536

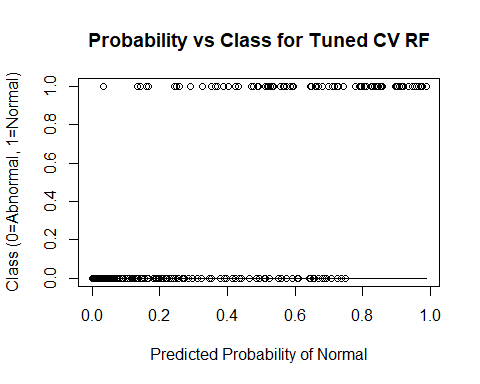
varImpPlot(random\_forest,type=2)



cv\_ranfor\_cf = caret::confusionMatrix( data2$classification, data2$pred\_class)  
cv\_ranfor\_cf

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Abnormal Normal  
## Abnormal 187 23  
## Normal 26 74  
##   
## Accuracy : 0.8419   
## 95% CI : (0.7965, 0.8807)  
## No Information Rate : 0.6871   
## P-Value [Acc > NIR] : 3.261e-10   
##   
## Kappa : 0.6355   
## Mcnemar's Test P-Value : 0.7751   
##   
## Sensitivity : 0.8779   
## Specificity : 0.7629   
## Pos Pred Value : 0.8905   
## Neg Pred Value : 0.7400   
## Prevalence : 0.6871   
## Detection Rate : 0.6032   
## Detection Prevalence : 0.6774   
## Balanced Accuracy : 0.8204   
##   
## 'Positive' Class : Abnormal   
##

plot(y=data2$class, x=data2$pred\_ranfor, main = "Probability vs Class for Tuned CV RF", xlab = "Predicted Probability of Normal", ylab="Class (0=Abnormal, 1=Normal)")  
lines(lowess(y = data2$class, x=data2$pred\_ranfor))



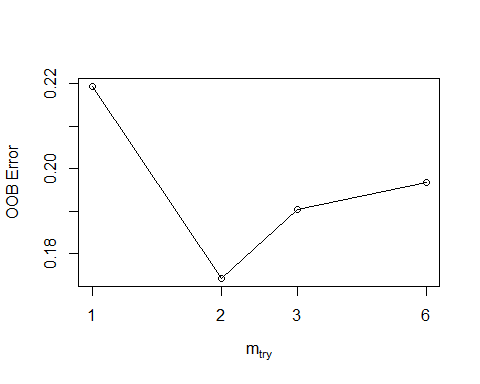
# Add Provided Random Noise

set.seed(12)  
df <- read.csv('Dataset\_spine.csv', col.names = c('pelvic\_incidence', 'pelvic\_tilt', 'lumbar\_lordosis\_angle', 'sacral\_slope', 'pelvic\_radius', 'degree\_spondylolisthesis', 'pelvic\_slope', 'Direct\_tilt', 'thoracic\_slope', 'cervical\_tilt', 'sacrum\_angle', 'scoliosis\_slope', 'classification', ' '))  
  
df$class = ifelse(df$classification == "Abnormal", 0, 1) #making classification numeric  
df$X. <- NULL #dropping the column with variable descriptions in it  
data = scale(df[, 1:12]) #scaling all except classification variable  
data = data.frame(data)  
data$classification = data1$classification #add classification back into scaled dataframe  
data$class = data1$class  
  
data$rand\_int = runif(n=nrow(data), min = 0, max = 1) #random values from uniform distribution  
data2 = data[order(data$rand\_int),] #reorder dataframe by random value  
  
  
data2$group = seq(from = 1, to=5, by=1)  
data2$pred\_ranfor = 0  
data2$pred\_class = as.factor(x = c("Abnormal", "Normal"))  
attach(data2)

## The following objects are masked \_by\_ .GlobalEnv:  
##   
## pred\_class, pred\_ranfor

im\_cv = tuneRF(x = data2[,1:12], y = data2[,13])

## mtry = 3 OOB error = 19.03%   
## Searching left ...  
## mtry = 2 OOB error = 17.42%   
## 0.08474576 0.05   
## mtry = 1 OOB error = 21.94%   
## -0.2592593 0.05   
## Searching right ...  
## mtry = 6 OOB error = 19.68%   
## -0.1296296 0.05



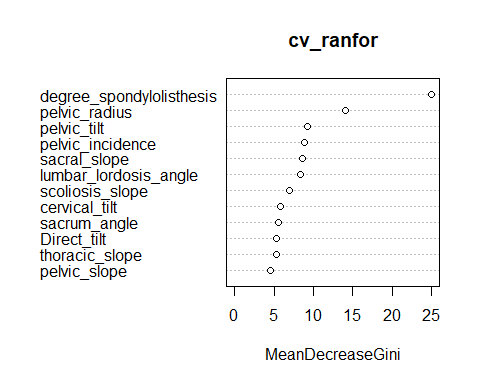
im\_cv

## mtry OOBError  
## 1.OOB 1 0.2193548  
## 2.OOB 2 0.1741935  
## 3.OOB 3 0.1903226  
## 6.OOB 6 0.1967742

for (grp in 1:5){  
 train = data2[data2$group != grp, ]  
 test = data2[data2$group == grp,]  
   
 cv\_ranfor <- randomForest(classification ~ . - group - rand\_int - class - pred\_ranfor - pred\_class, data = train, mtry=2)  
  
 pred\_ranfor = predict(object = cv\_ranfor, test, "prob")[,"Normal"]  
 data2$pred\_ranfor[data2$group == grp] = pred\_ranfor  
   
 pred\_class = predict(object = cv\_ranfor, test)  
 data2$pred\_class[data2$group == grp] = pred\_class  
}  
  
importance(cv\_ranfor)

## MeanDecreaseGini  
## pelvic\_incidence 8.859955  
## pelvic\_tilt 9.250360  
## lumbar\_lordosis\_angle 8.412710  
## sacral\_slope 8.561810  
## pelvic\_radius 14.124541  
## degree\_spondylolisthesis 25.057467  
## pelvic\_slope 4.515013  
## Direct\_tilt 5.361990  
## thoracic\_slope 5.314960  
## cervical\_tilt 5.856255  
## sacrum\_angle 5.578070  
## scoliosis\_slope 6.937531

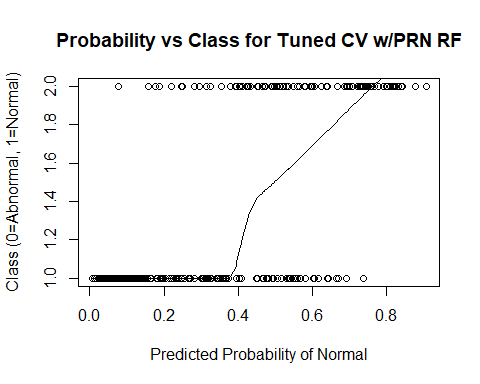
varImpPlot(cv\_ranfor,type=2)



cv\_ranfor\_cf\_wpn = caret::confusionMatrix( data2$classification, data2$pred\_class)  
cv\_ranfor\_cf\_wpn

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Abnormal Normal  
## Abnormal 188 22  
## Normal 35 65  
##   
## Accuracy : 0.8161   
## 95% CI : (0.7684, 0.8577)  
## No Information Rate : 0.7194   
## P-Value [Acc > NIR] : 5.282e-05   
##   
## Kappa : 0.5645   
## Mcnemar's Test P-Value : 0.112   
##   
## Sensitivity : 0.8430   
## Specificity : 0.7471   
## Pos Pred Value : 0.8952   
## Neg Pred Value : 0.6500   
## Prevalence : 0.7194   
## Detection Rate : 0.6065   
## Detection Prevalence : 0.6774   
## Balanced Accuracy : 0.7951   
##   
## 'Positive' Class : Abnormal   
##

plot(y=data2$class, x=data2$pred\_ranfor, main = "Probability vs Class for Tuned CV w/PRN RF", xlab = "Predicted Probability of Normal", ylab="Class (0=Abnormal, 1=Normal)")  
lines(lowess(y = data2$class, x=data2$pred\_ranfor))

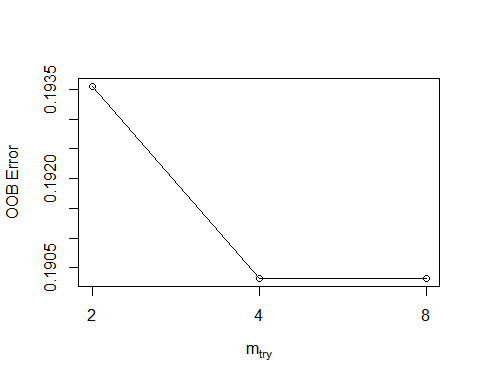


# Add 10 Random Variables

df <- read.csv('Dataset\_spine.csv', col.names = c('pelvic\_incidence', 'pelvic\_tilt', 'lumbar\_lordosis\_angle', 'sacral\_slope', 'pelvic\_radius', 'degree\_spondylolisthesis', 'pelvic\_slope', 'Direct\_tilt', 'thoracic\_slope', 'cervical\_tilt', 'sacrum\_angle', 'scoliosis\_slope', 'classification', ' '))  
set.seed(12)  
df$class = ifelse(df$classification == "Abnormal", 0, 1)  
df$X. <- NULL #dropping the column with variable descriptions in it  
data = scale(df[, 1:12]) #scaling all except classification variable  
data = data.frame(data)  
data$classification = df$classification #add classification back into scaled dataframe  
data$class = df$class  
  
rand\_df = data.frame(matrix(rnorm(10\*nrow(data)), nrow = nrow(data), ncol = 10))  
data = cbind(data, rand\_df)  
#write.csv(x = data, file = "data\_with10.csv")  
  
  
  
data$rand\_int = runif(n=nrow(data), min = 0, max = 1) #random values from uniform distribution  
data2 = data[order(data$rand\_int),] #reorder dataframe by random value  
  
  
data2$group = seq(from = 1, to=5, by=1)  
data2$pred\_ranfor = 0  
data2$pred\_class = as.factor(x = c("Abnormal", "Normal"))  
attach(data2)

im\_cv = tuneRF(x = data2[,c(1:12,15:24)], y = data2[,13])

## mtry = 4 OOB error = 19.03%   
## Searching left ...  
## mtry = 2 OOB error = 19.35%   
## -0.01694915 0.05   
## Searching right ...  
## mtry = 8 OOB error = 19.03%   
## 0 0.05



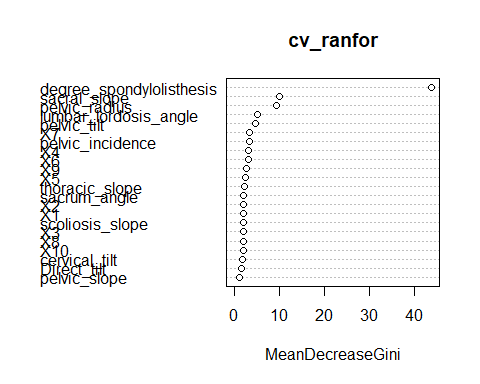
im\_cv

## mtry OOBError  
## 2.OOB 2 0.1935484  
## 4.OOB 4 0.1903226  
## 8.OOB 8 0.1903226

for (grp in 1:5){  
 train = data2[data2$group != grp, ]  
 test = data2[data2$group == grp,]  
   
 cv\_ranfor <- randomForest(classification ~ . - group - rand\_int - class - pred\_ranfor - pred\_class, data = train, mtry=16)  
  
 pred\_ranfor = predict(object = cv\_ranfor, test, "prob")[,"Normal"]  
 data2$pred\_ranfor[data2$group == grp] = pred\_ranfor  
   
 pred\_class = predict(object = cv\_ranfor, test)  
 data2$pred\_class[data2$group == grp] = pred\_class  
}  
  
importance(cv\_ranfor)

## MeanDecreaseGini  
## pelvic\_incidence 3.2022659  
## pelvic\_tilt 4.5670598  
## lumbar\_lordosis\_angle 5.0248268  
## sacral\_slope 9.9702033  
## pelvic\_radius 9.1985254  
## degree\_spondylolisthesis 43.9117143  
## pelvic\_slope 0.9751303  
## Direct\_tilt 1.4061935  
## thoracic\_slope 2.0743864  
## cervical\_tilt 1.8274781  
## sacrum\_angle 1.9724421  
## scoliosis\_slope 1.8839118  
## X1 1.9651829  
## X2 1.9712072  
## X3 1.8640025  
## X4 3.1620878  
## X5 2.3345345  
## X6 3.1114970  
## X7 3.3487048  
## X8 1.8452842  
## X9 2.6035675  
## X10 1.8380684

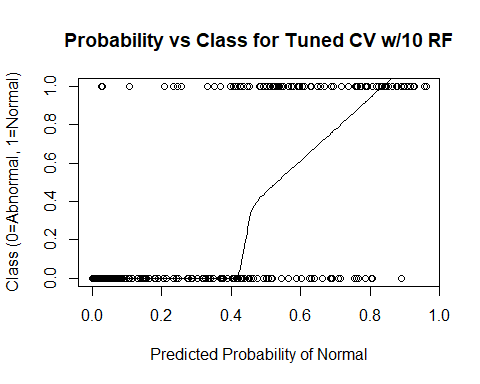
varImpPlot(cv\_ranfor,type=2)



cv\_ranfor\_cf\_10 = caret::confusionMatrix( data2$classification, data2$pred\_class)  
cv\_ranfor\_cf\_10

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Abnormal Normal  
## Abnormal 185 25  
## Normal 25 75  
##   
## Accuracy : 0.8387   
## 95% CI : (0.7929, 0.8779)  
## No Information Rate : 0.6774   
## P-Value [Acc > NIR] : 8.779e-11   
##   
## Kappa : 0.631   
## Mcnemar's Test P-Value : 1   
##   
## Sensitivity : 0.8810   
## Specificity : 0.7500   
## Pos Pred Value : 0.8810   
## Neg Pred Value : 0.7500   
## Prevalence : 0.6774   
## Detection Rate : 0.5968   
## Detection Prevalence : 0.6774   
## Balanced Accuracy : 0.8155   
##   
## 'Positive' Class : Abnormal   
##

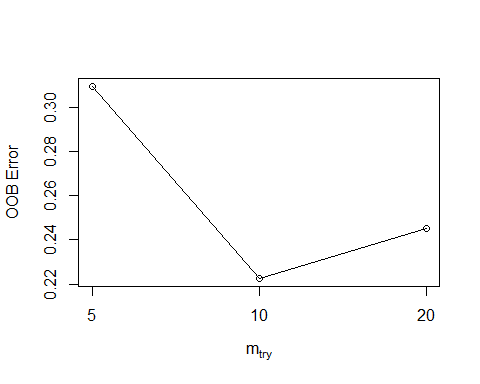
plot(y=data2$class, x=data2$pred\_ranfor, main = "Probability vs Class for Tuned CV w/10 RF", xlab = "Predicted Probability of Normal", ylab="Class (0=Abnormal, 1=Normal)")  
lines(lowess(y = data2$class, x=data2$pred\_ranfor))

 # Add 100 Random Variables

df <- read.csv('Dataset\_spine.csv', col.names = c('pelvic\_incidence', 'pelvic\_tilt', 'lumbar\_lordosis\_angle', 'sacral\_slope', 'pelvic\_radius', 'degree\_spondylolisthesis', 'pelvic\_slope', 'Direct\_tilt', 'thoracic\_slope', 'cervical\_tilt', 'sacrum\_angle', 'scoliosis\_slope', 'classification', ' '))  
set.seed(12)  
df$class = ifelse(df$classification == "Abnormal", 0, 1)  
df$X. <- NULL #dropping the column with variable descriptions in it  
data = scale(df[, 1:12]) #scaling all except classification variable  
data = data.frame(data)  
data$classification = df$classification #add classification back into scaled dataframe  
data$class = df$class  
  
rand\_df = data.frame(matrix(rnorm(100\*nrow(data)), nrow = nrow(data), ncol = 100))  
data = cbind(data, rand\_df)  
#write.csv(x = data, file = "data\_with100.csv")  
  
  
  
data$rand\_int = runif(n=nrow(data), min = 0, max = 1) #random values from uniform distribution  
data2 = data[order(data$rand\_int),] #reorder dataframe by random value  
  
  
data2$group = seq(from = 1, to=5, by=1)  
data2$pred\_ranfor = 0  
data2$pred\_class = as.factor(x = c("Abnormal", "Normal"))  
attach(data2)

im\_cv = tuneRF(x = data2[,c(1:12,15:114)], y = data2[,13])

## mtry = 10 OOB error = 22.26%   
## Searching left ...  
## mtry = 5 OOB error = 30.97%   
## -0.3913043 0.05   
## Searching right ...  
## mtry = 20 OOB error = 24.52%   
## -0.1014493 0.05

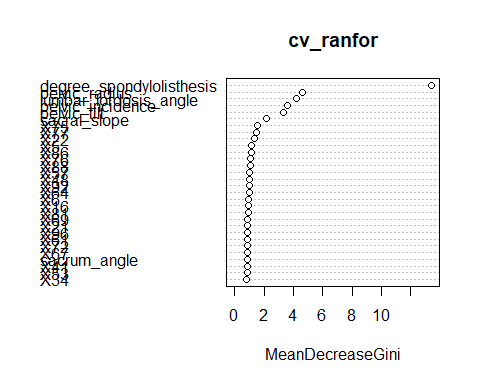


im\_cv

## mtry OOBError  
## 5.OOB 5 0.3096774  
## 10.OOB 10 0.2225806  
## 20.OOB 20 0.2451613

for (grp in 1:5){  
 train = data2[data2$group != grp, ]  
 test = data2[data2$group == grp,]  
   
 cv\_ranfor <- randomForest(classification ~ . - group - rand\_int - class - pred\_ranfor - pred\_class, data = train, mtry=10)  
  
 pred\_ranfor = predict(object = cv\_ranfor, test, "prob")[,"Normal"]  
 data2$pred\_ranfor[data2$group == grp] = pred\_ranfor  
   
 pred\_class = predict(object = cv\_ranfor, test)  
 data2$pred\_class[data2$group == grp] = pred\_class  
}  
  
importance(cv\_ranfor)

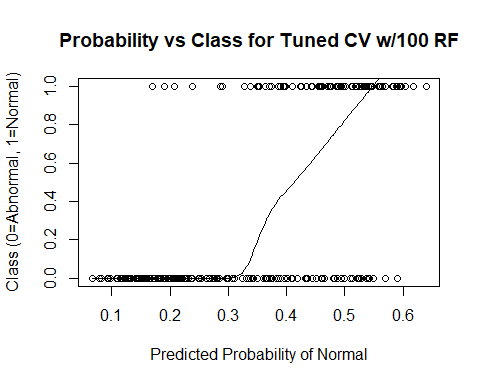
## MeanDecreaseGini  
## pelvic\_incidence 3.5641762  
## pelvic\_tilt 3.3364984  
## lumbar\_lordosis\_angle 4.1795320  
## sacral\_slope 2.1758695  
## pelvic\_radius 4.6062258  
## degree\_spondylolisthesis 13.4208132  
## pelvic\_slope 0.5021879  
## Direct\_tilt 0.4562610  
## thoracic\_slope 0.6121050  
## cervical\_tilt 0.6490038  
## sacrum\_angle 0.8462769  
## scoliosis\_slope 0.8053812  
## X1 0.6778439  
## X2 0.7681018  
## X3 0.5986865  
## X4 0.6395936  
## X5 1.1574296  
## X6 0.9578584  
## X7 0.8195534  
## X8 0.5693077  
## X9 0.8036344  
## X10 0.5131597  
  
varImpPlot(cv\_ranfor,type=2)



cv\_ranfor\_cf\_100 = caret::confusionMatrix( data2$classification, data2$pred\_class)  
cv\_ranfor\_cf\_100

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Abnormal Normal  
## Abnormal 192 18  
## Normal 58 42  
##   
## Accuracy : 0.7548   
## 95% CI : (0.703, 0.8017)  
## No Information Rate : 0.8065   
## P-Value [Acc > NIR] : 0.9896   
##   
## Kappa : 0.3734   
## Mcnemar's Test P-Value : 7.691e-06   
##   
## Sensitivity : 0.7680   
## Specificity : 0.7000   
## Pos Pred Value : 0.9143   
## Neg Pred Value : 0.4200   
## Prevalence : 0.8065   
## Detection Rate : 0.6194   
## Detection Prevalence : 0.6774   
## Balanced Accuracy : 0.7340   
##   
## 'Positive' Class : Abnormal   
##

plot(y=data2$class, x=data2$pred\_ranfor, main = "Probability vs Class for Tuned CV w/100 RF", xlab = "Predicted Probability of Normal", ylab="Class (0=Abnormal, 1=Normal)")  
lines(lowess(y = data2$class, x=data2$pred\_ranfor))



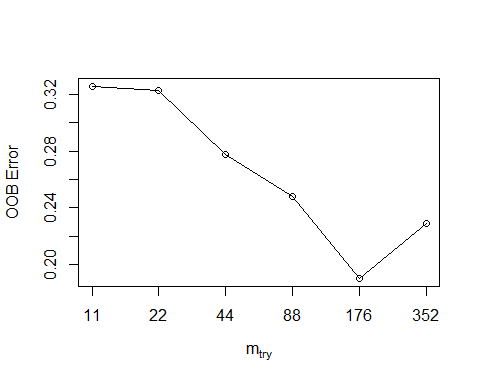
# Add 500 Random Variables

df <- read.csv('Dataset\_spine.csv', col.names = c('pelvic\_incidence', 'pelvic\_tilt', 'lumbar\_lordosis\_angle', 'sacral\_slope', 'pelvic\_radius', 'degree\_spondylolisthesis', 'pelvic\_slope', 'Direct\_tilt', 'thoracic\_slope', 'cervical\_tilt', 'sacrum\_angle', 'scoliosis\_slope', 'classification', ' '))  
set.seed(12)  
df$class = ifelse(df$classification == "Abnormal", 0, 1)  
df$X. <- NULL #dropping the column with variable descriptions in it  
data = scale(df[, 1:12]) #scaling all except classification variable  
data = data.frame(data)  
data$classification = df$classification #add classification back into scaled dataframe  
data$class = df$class  
  
rand\_df = data.frame(matrix(rnorm(500\*nrow(data)), nrow = nrow(data), ncol = 500))  
data = cbind(data, rand\_df)  
#write.csv(x = data, file = "data\_with500.csv")  
  
  
  
data$rand\_int = runif(n=nrow(data), min = 0, max = 1) #random values from uniform distribution  
data2 = data[order(data$rand\_int),] #reorder dataframe by random value  
  
  
data2$group = seq(from = 1, to=5, by=1)  
data2$pred\_ranfor = 0  
data2$pred\_class = as.factor(x = c("Abnormal", "Normal"))  
attach(data2)

## The following objects are masked \_by\_ .GlobalEnv:  
##   
## pred\_class, pred\_ranfor

im\_cv = tuneRF(x = data2[,c(1:12,15:514)], y = data2[,13])

## mtry = 22 OOB error = 32.26%   
## Searching left ...  
## mtry = 11 OOB error = 32.58%   
## -0.01 0.05   
## Searching right ...  
## mtry = 44 OOB error = 27.74%   
## 0.14 0.05   
## mtry = 88 OOB error = 24.84%   
## 0.1046512 0.05   
## mtry = 176 OOB error = 19.03%   
## 0.2337662 0.05   
## mtry = 352 OOB error = 22.9%   
## -0.2033898 0.05

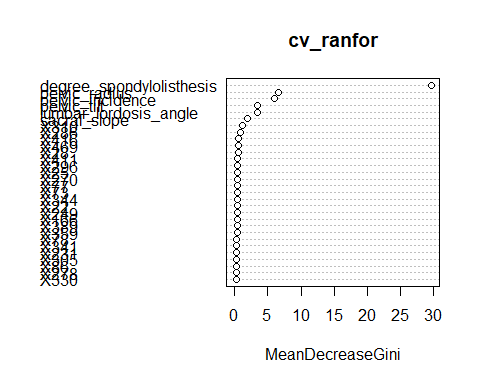


im\_cv

## mtry OOBError  
## 11.OOB 11 0.3258065  
## 22.OOB 22 0.3225806  
## 44.OOB 44 0.2774194  
## 88.OOB 88 0.2483871  
## 176.OOB 176 0.1903226  
## 352.OOB 352 0.2290323

for (grp in 1:5){  
 train = data2[data2$group != grp, ]  
 test = data2[data2$group == grp,]  
   
 cv\_ranfor <- randomForest(classification ~ . - group - rand\_int - class - pred\_ranfor - pred\_class, data = train, mtry=176)  
  
 pred\_ranfor = predict(object = cv\_ranfor, test, "prob")[,"Normal"]  
 data2$pred\_ranfor[data2$group == grp] = pred\_ranfor  
   
 pred\_class = predict(object = cv\_ranfor, test)  
 data2$pred\_class[data2$group == grp] = pred\_class  
}  
  
importance(cv\_ranfor)

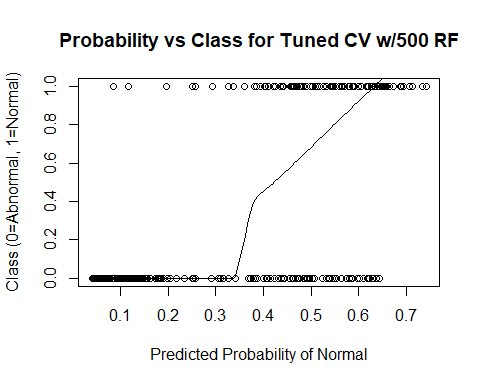
## MeanDecreaseGini  
## pelvic\_incidence 6.01100462  
## pelvic\_tilt 3.38621535  
## lumbar\_lordosis\_angle 3.38204323  
## sacral\_slope 1.84095368  
## pelvic\_radius 6.61375200  
## degree\_spondylolisthesis 29.69157311  
## pelvic\_slope 0.04516613  
## Direct\_tilt 0.05921861  
## thoracic\_slope 0.28618526  
## cervical\_tilt 0.10897747  
## sacrum\_angle 0.05223586  
## scoliosis\_slope 0.06663832  
## X1 0.02960129  
## X2 0.07636687  
## X3 0.16661184  
## X4 0.04863451  
## X5 0.10742465  
## X6 0.26578690  
## X7 0.17616101  
## X8 0.08328187  
## X9 0.19831939  
## X10 0.04033133  
## X11 0.11273203  
## X12 0.06154676  
## X13 0.12715521  
## X14 0.21590919  
## X15 0.01675745  
## X16 0.07298429  
## X17 0.09978005  
## X18 0.07644006  
## X19 0.04767568  
## X20 0.01928116  
varImpPlot(cv\_ranfor,type=2)



cv\_ranfor\_cf\_500 = caret::confusionMatrix( data2$classification, data2$pred\_class)  
cv\_ranfor\_cf\_500

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Abnormal Normal  
## Abnormal 182 28  
## Normal 37 63  
##   
## Accuracy : 0.7903   
## 95% CI : (0.7407, 0.8343)  
## No Information Rate : 0.7065   
## P-Value [Acc > NIR] : 0.0005294   
##   
## Kappa : 0.5087   
## Mcnemar's Test P-Value : 0.3210620   
##   
## Sensitivity : 0.8311   
## Specificity : 0.6923   
## Pos Pred Value : 0.8667   
## Neg Pred Value : 0.6300   
## Prevalence : 0.7065   
## Detection Rate : 0.5871   
## Detection Prevalence : 0.6774   
## Balanced Accuracy : 0.7617   
##   
## 'Positive' Class : Abnormal   
##

plot(y=data2$class, x=data2$pred\_ranfor, main = "Probability vs Class for Tuned CV w/500 RF", xlab = "Predicted Probability of Normal", ylab="Class (0=Abnormal, 1=Normal)")  
lines(lowess(y = data2$class, x=data2$pred\_ranfor))

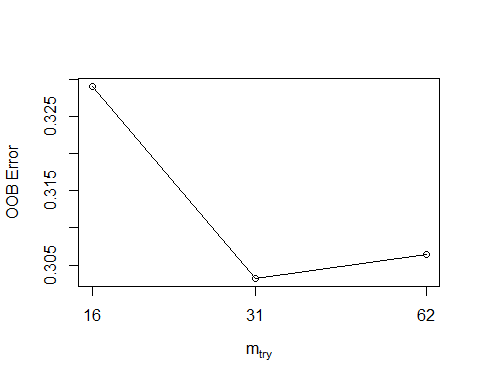


# Add 1000 Random Variables

df <- read.csv('Dataset\_spine.csv', col.names = c('pelvic\_incidence', 'pelvic\_tilt', 'lumbar\_lordosis\_angle', 'sacral\_slope', 'pelvic\_radius', 'degree\_spondylolisthesis', 'pelvic\_slope', 'Direct\_tilt', 'thoracic\_slope', 'cervical\_tilt', 'sacrum\_angle', 'scoliosis\_slope', 'classification', ' '))  
set.seed(12)  
df$class = ifelse(df$classification == "Abnormal", 0, 1)  
df$X. <- NULL #dropping the column with variable descriptions in it  
data = scale(df[, 1:12]) #scaling all except classification variable  
data = data.frame(data)  
data$classification = df$classification #add classification back into scaled dataframe  
data$class = df$class  
  
rand\_df = data.frame(matrix(rnorm(1000\*nrow(data)), nrow = nrow(data), ncol = 1000))  
data = cbind(data, rand\_df)  
#write.csv(x = data, file = "data\_with1000.csv")  
  
  
  
data$rand\_int = runif(n=nrow(data), min = 0, max = 1) #random values from uniform distribution  
data2 = data[order(data$rand\_int),] #reorder dataframe by random value  
  
data2$group = seq(from = 1, to=5, by=1)  
data2$pred\_ranfor = 0  
data2$pred\_class = as.factor(x = c("Abnormal", "Normal"))  
attach(data2)

im\_cv = tuneRF(x = data2[,c(1:12,15:1014)], y = data2[,13])

## mtry = 31 OOB error = 30.32%   
## Searching left ...  
## mtry = 16 OOB error = 32.9%   
## -0.08510638 0.05   
## Searching right ...  
## mtry = 62 OOB error = 30.65%   
## -0.0106383 0.05

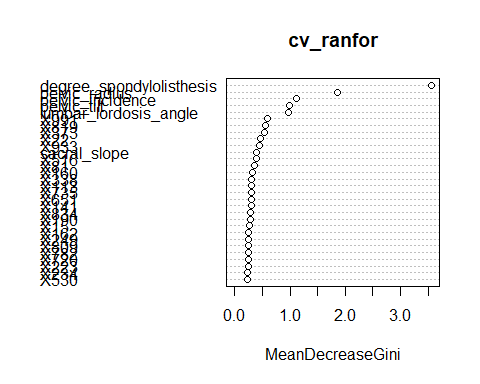


im\_cv

## mtry OOBError  
## 16.OOB 16 0.3290323  
## 31.OOB 31 0.3032258  
## 62.OOB 62 0.3064516

for (grp in 1:5){  
 train = data2[data2$group != grp, ]  
 test = data2[data2$group == grp,]  
   
 cv\_ranfor <- randomForest(classification ~ . - group - rand\_int - class - pred\_ranfor - pred\_class, data = train, mtry=31)  
  
 pred\_ranfor = predict(object = cv\_ranfor, test, "prob")[,"Normal"]  
 data2$pred\_ranfor[data2$group == grp] = pred\_ranfor  
   
 pred\_class = predict(object = cv\_ranfor, test)  
 data2$pred\_class[data2$group == grp] = pred\_class  
}  
  
importance(cv\_ranfor)

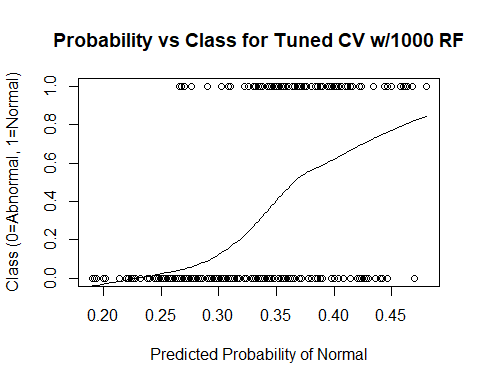
## MeanDecreaseGini  
## pelvic\_incidence 1.115865888  
## pelvic\_tilt 0.981450644  
## lumbar\_lordosis\_angle 0.975049651  
## sacral\_slope 0.395121701  
## pelvic\_radius 1.849761856  
## degree\_spondylolisthesis 3.565151873  
## pelvic\_slope 0.043054130  
## Direct\_tilt 0.112771258  
## thoracic\_slope 0.036546526  
## cervical\_tilt 0.048519792  
## sacrum\_angle 0.084515479  
## scoliosis\_slope 0.098642966  
## X1 0.110725066  
## X2 0.095204713  
## X3 0.084426512  
## X4 0.133966330  
## X5 0.196215418  
## X6 0.066426525  
## X7 0.101167765  
## X8 0.102328764  
## X9 0.086409782  
## X10 0.060620587  
## X11 0.119566181  
## X12 0.138858466  
## X13 0.045334508  
## X14 0.071295411  
## X15 0.259547404  
## X16 0.065672330  
## X17 0.069879710  
## X18 0.112287686  
## X19 0.088829552  
## X20 0.066341833  
## X21 0.081837614  
## X22 0.473760239  
## X23 0.070109449  
## X24 0.050637532  
## X25 0.086569566  
## X26 0.117118955  
## X27 0.106494432  
## X28 0.140308903  
## X29 0.062346060  
## X30 0.080721929  
## X31 0.062293997  
## X32 0.102054121  
## X33 0.043153305  
## X34 0.130603745  
## X35 0.095472731  
## X36 0.118973340  
## X37 0.093745077  
## X38 0.075535333  
## X39 0.037310434  
## X40 0.078961066  
varImpPlot(cv\_ranfor,type=2)



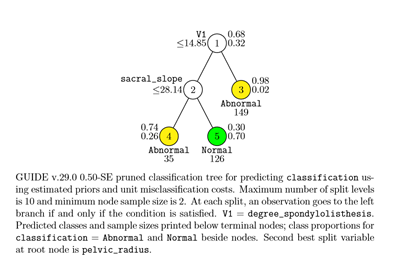
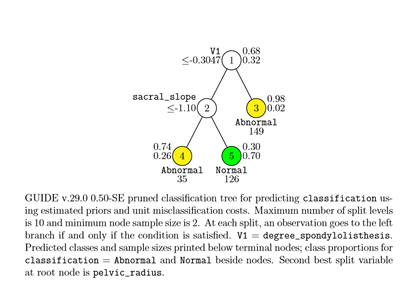
cv\_ranfor\_cf\_1000 = caret::confusionMatrix( data2$classification, data2$pred\_class)  
cv\_ranfor\_cf\_1000

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Abnormal Normal  
## Abnormal 210 0  
## Normal 100 0  
##   
## Accuracy : 0.6774   
## 95% CI : (0.6223, 0.7292)  
## No Information Rate : 1   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.6774   
## Specificity : NA   
## Pos Pred Value : NA   
## Neg Pred Value : NA   
## Prevalence : 1.0000   
## Detection Rate : 0.6774   
## Detection Prevalence : 0.6774   
## Balanced Accuracy : NA   
##   
## 'Positive' Class : Abnormal   
##

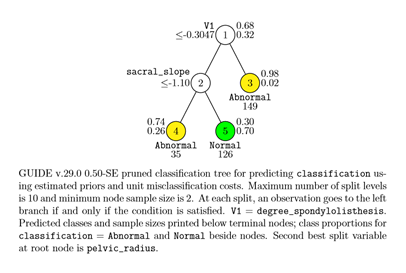
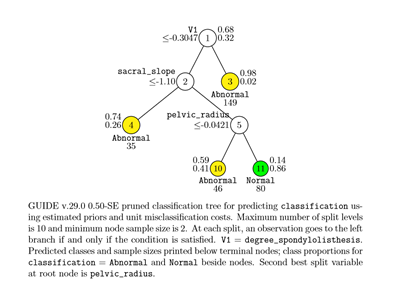
plot(y=data2$class, x=data2$pred\_ranfor, main = "Probability vs Class for Tuned CV w/1000 RF", xlab = "Predicted Probability of Normal", ylab="Class (0=Abnormal, 1=Normal)")  
lines(lowess(y = data2$class, x=data2$pred\_ranfor))

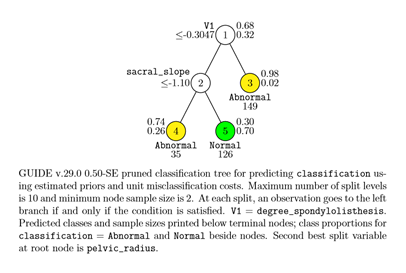


Appendix V: GUIDE Tree Diagrams

1) Tree for original data 2) Tree for data with 10 Random Variables

3) Tree with 100 Random Variables 4) Tree with 500 Random Variables



5) Tree diagram with 1000 Random Variables  


Sources

Breiman, Leo, et al. “Package ‘RandomForest.’” r-Project, 25 Mar. 2018.

Hunt, Tyler. “Package ‘ModelMetrics.’” r-Project, 3 Nov. 2018.

Kuhn, Max. “Package ‘Caret.’” r-Project, 27 May 2018.

Loh, Wei-Yin. “GUIDE User Manual.” Department of Statistics, University of Wisconsin–Madison, 18 Oct. 2018.

Meyer, David, et al. “Package ‘e1071.’” r-Project, 28 July 2018.

Ng, Andrew. “Support Vector Machines.” cs229, Stanford.edu.

“Random Forests Leo Breiman and Adele Cutler.” Statistics at UC Berkeley | Department of Statistics.